An Allele Sharing Method for Fine Mapping Linkage Loci: Application to Bipolar Affective Disorder

Andrew J. Lee

A thesis submitted for the degree of Ph.D.

University of Edinburgh

2009

Main Contents

Contents of Figures

Contents of Tables

Declaration

I hereby declare that all work in this thesis, unless otherwise stated, was carried out by the candidate and has not been submitted for any other degree or professional qualification. I further declare that the thesis was composed by the candidate.

Andrew Lee

Acknowledgements

This work was funded by the UK Medical Research Council and conducted at the University of Edinburgh's Medical Genetics Section. I would like to thank my supervisors, Prof. David Porteous, Dr. Kathryn Evans and Dr. Richard Adams.

I would also like to thank Dr. Naomi Wray and Dr. Pippa Thompson for their guidance and support; all the staff in the Medical Genetics Section who contributed to this work; Prof. Douglas Blackwood, Dr. Walter Muir, Dr. Claude Férec, Dr. Sevilla Detera-Wadleigh, Prof. Michael Owen and Dr. Philip Asherson for access to the families studied in this work.

Finally, I would like to thank my family and friends who have supported me throughout this work.

Abbreviations

BP	Bipolar
BPAD	Bipolar affective disorder
Chr.	Chromosome
CEPH	Centre d'Etudes du Polymorphisme Humain
CEU	CEPH from Utah
CF	Cystic fibrosis
CFTR	Cystic fibrosis transmembrane conductance regulator
cM	Centi-morgan
CNV	Copy number variant
DNA	Deoxyribonucleic Acid
FBAT	Familyibased association test
GUI	Graphical user interface
HLA	Humanleukocyte antigen
HSS	Haplotype sharing statistic
htSNP	Haplotype tagging SNP
kb	Kilobases
LAMP	Linkage and association modelling in pedigrees
LOD	Log of odds
LD	Linkage disequilibrium
MAF	Minor-allele frequency
Mb	Megabases
MIM	Mendelian Inheritance in Man
mRNA	Messenger RNA
MILC	Maximum identity length contrast
NCBI	National Center for Biotechnology Information
OR	Odds ratio
RefSeq	NCBI reference sequence
RNA	Ribonucleic acid
SAM	Schizoaffective disorder
SNP	Single nucleotide polymorphism
SCZ	Schizophrenia
STR	Short tandem repeat
TDT	Transmission disequilibrium test
TDTae	Transmission disequilibrium test with allowance for errors
WHAP	Weighted Haplotype Program

Abstract

An Allele Sharing Method for Fine Mapping Linkage Loci: Application to Bipolar Affective Disorder

Large family studies of complex disorders can be used to detect a genomic region linked with a particular illness. Where multiple families are found with common regions of linkage, this could be due to an ancestral mutation common to these families. In this thesis, I describe a method for studying allele sharing in families that share a linkage region, to identify a common founder mutation, thus maximising the results of replicated linkage studies.

The method tests the hypothesis that the evidence for shared linkage is derived from the sharing of a common affected ancestor. By comparing the allelic similarity of haplotypes across common linkage regions, it is possible to identify any regions that are identical by descent between the families. A method of permutation analysis followed by a nested permutation technique have been developed to assess the statistical significance of allele sharing scores. Chapter 3 describes the proof of principle of the method through its application to known cystic fibrosis mutations and through simulated datasets. This provides both a real dataset and a much more diverse range of simulated conditions on which to test the method. The range of simulated data was also used to develop a set of criteria for the effective us of the method.

In Chapter 4, the allele sharing method was applied to two replicated linkage regions on chromosome 4p15-16 that segregate with bipolar affective disorder. This was done over two phases, first taking in markers covering the genic regions of the shared linkage region and then followed up with a complete coverage of the region. This analysis identified a 200kb region with significant confidence within the 8Mb of the two linkage regions. The study of this region presents a clear example of how replicated linkage results that are caused by some founder effect, can be examined, and refined using this allele sharing method to vastly reduce the region under investigation.

Chapter 1

Introduction

Large families with multiple members affected with a specific disorder can be used to detect genomic regions linked to that disorder, however the regions identified are usually large. One of the challenges facing those tackling the genetic analysis of complex disease is how best to capitalise on linkage results, without having to scan the entire region for association. If multiple families with overlapping linkage regions exist, then these regions may harbour a common ancestral mutation. It has been shown that a region of the ancestral haplotype flanking such a mutation should be expected to be found in linked families. A new method is required to take advantage of advances in genotyping technology to allow the search for these ancestral regions of shared haplotype. Such a method could provide an invaluable tool in bridging the gap between genome wide linkage studies and the identification of a disease gene.

Bipolar affective disorder (BPAD) presents a complex genetic etiology and large family studies have been used to generate a number of linkage signals across the genome. Yet none of these have led to the identification of a causal variant. In particular, a number of families have been shown to display linkage to the chromosome 4p15-16 region. While there are many genes in the region that could be implicated in BPAD, there is no stand-out functional candidate gene in the region on which to focus further research. A new method to reduce the region under investigation could prove invaluable in this study of BPAD, as well as in similar studied of other complex disorders.

1.1 Finding disease genes

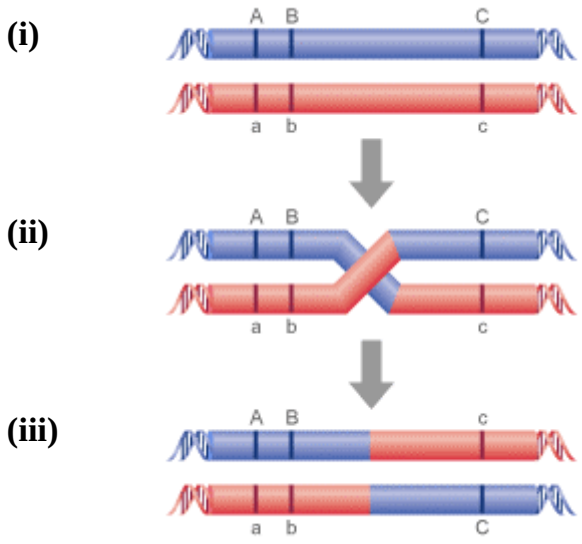
Genetic and environmental factors have a role in the development of any disease. Finding the genes that are responsible for causing susceptibility to illness is important in a number of areas: (i) to help understand the pathophysiology of disease; (ii) to help in understanding the biology of specific systems or organs; (iii) to improve the diagnosis of disease and (iv) to aid the identification of drug targets. The development of linkage studies using genetic markers has revolutionised gene discovery and genetic linkage, or positional cloning, is now commonly used to try and identify the genes that are responsible for, or that contribute to, the development of disease. Positional cloning identifies a disease gene solely using information about its chromosomal location. The basic principles of positional cloning are to first map the disease as finely as possible in affected families, then identify candidate genes and, finally, detect mutations of these genes in patients.

1.1.1 Genetic linkage

The purpose of a linkage study is to identify a chromosomal region that contains a susceptibility gene by studying the pattern of inheritance within families. This is done in three stages, (i) the clinical assessment of the subjects to determine a specific phenotype; (ii) testing genetic material from the subjects at a series of genetic markers and (iii) using statistical techniques to test whether a specific phenotype and genetic markers segregate independently or together (cosegregate). If they cosegregate, it is assumed that the genetic marker is located close to the gene that is causing susceptibility to the phenotype. The principle underlying this is that the further away a genetic marker is from the disease causing gene, the greater the chance of a recombination event taking place between the two and therefore the chances of cosegregation decreases (see Figure 1.1). This can typically identify a region harbouring a disease gene to around 20cM if sufficient number of meioses are available (e.g. Hu et al. 2000, Wiesner et al. 2003, Koskenmies et al. 2004, Duggal et al. 2007). LOD (log of odds) scores are usually used to measure the probability of linkage and calculate the likelihood of attaining the observed results based on an assumption of linkage compared with the likelihood based on chance alone. See Ott (1999) for a detailed discussion of genetic linkage. In this way, linkage has been used

to help identify many genes that are linked to disease although there has been less success in its application to complex disorders. This is discussed in section 1.1.4. Throughout this thesis, the term 'linkage region' is used to refer to these genomic locations that are found to be of interest through these linkage studies. Such regions are defined by the chromosomal location that contains those markers (or marker) that have been shown to cosegregate with a trait or disease and are flanked by those makers that are shown to segregate independently. The generally accepted definition of a significant region of genome-wide linkage, as provided by Lander & Kruglyak (1995) was of a LOD score of 3.3 for primary evidence. Where primary evidence of linkage exists, additional studies showing LOD score of 1.9 should be taken as evidence of replication.

Figure 1.1: Principle of linkage analysis.



(i) *shows two parental chromosomes aligned in a germ cell during meiosis. A, B and C represent three different genetic locations.* ***(ii)*** *shows how recombination can occur, involving the crossing over of DNA strands.* ***(iii)*** *shows the resulting chromosomes. If, for example, A is a disease gene and B and C are genetic markers, recombination is more likely to occur between A and C than A and B and marker B is therefore more likely to cosegregate with the disease. Image reproduced from Twyman (2003).*

1.1.2 Genetic association

Genetic association is defined as the non-random association of two or more traits, where at least one of these is genetic. Genetic association studies aim to detect association between one or more genetic polymorphisms and a disease or trait. An association exists if an allele, genotype or haplotype is seen more often than expected by chance in individuals presenting the trait. Therefore an individual carrying the relevant allele, genotype or haplotype is at an increased risk of presenting the associated trait. Genetic association is closely linked to the concept of genetic linkage, differing in that association requires the same allele to be associated with the disease in a similar way across a population, while linkage allows for different alleles to be associated with the disease in different families. However, it has been argued that, as genetic associations are due to common ancestry in a population, they are just a special case of linkage in which the population can be considered as the extended family. In linkage analysis, smaller regions are more likely to be detected by studying distantly related individuals than closely related individuals, although a higher density of markers is required as the linkage will extend over a shorter region, due to greater genetic recombination in more distant relatives. It could be argued that association between apparently unrelated individuals represents the ultimate extension of this effect, leading association studies to have greater power than linkage studies, but requiring many more markers to be studied. The association itself can be due to a direct or indirect effect, or be due to confounding. Direct association refers to the case of a putative causal variant while indirect association refers to the association of locus that is a proxy for the causal locus. Indirect association allows association studies that don't need to have a candidate variant, allowing a candidate region to be tested. Confounding can be caused by features of the test population such as population stratification or admixture.

The mapping of susceptibility genes for complex disorders by indirect association relies on the existence of association between causal variants and nearby markers at the population level. Such an association is referred to as Linkage Disequilibrium (LD). LD can also be used more generally to refer to any situation where some combinations of alleles occur more or less frequently in a population than would be expected if the loci were segregating independently even when not due to association.

Non-random associations between alleles at different loci are measured by the degree of linkage disequilibrium. Unless some countervailing process maintains it, LD is expected to vary at a rate that is the inverse of the local recombination rate (Nachman et al. 2002). The appearance of significant LD is generally found in natural populations for genes that are tightly linked or for genes that are within or near an inverted segment of chromosome. Significant LD can also result from admixture of two or more sub-populations with differing allele frequencies. LD may also be caused by inbreeding.

There are two main types of association study, population-based, or family-based. These both have different strengths and weaknesses and should be viewed as complimentary both to each other and to other methods such as genetic linkage studies. Population based, or case-control, studies generally compare the frequency of alleles or genotypes in subjects carrying a disease or trait (cases) with those who don't (controls; randomly selected from the same population). Where there is a difference in allele or genotype frequency between the two groups, this would indicate that the allele or genotype in question is associated with the disease or trait either directly or indirectly. Family-based designs generally use the parents of affected individuals as the controls. These types of tests tend to compare the alleles that were transmitted from parents to affected offspring against those that were not transmitted. So if an allele or genotype is transmitted more often than expected in equal transmission then that allele or genotype is regarded as showing association with the disease in question. The most common example of this is the transmission disequilibrium test (TDT; Speilman et al. 1993). TDT was first proposed in the context of family trios where it tests the transmission of alleles from heterozygous parents to affected offspring. There have since been many similar methods published that provide some extension to the original method, such as those based on analysis of large families (Martin et al. 2000), multi-allelic markers (Cucca & Todd 1996) and those that include covariate information (Lunetta et al. 2000). The advantages and disadvantages of some of these methods will be discussed in more detail below. Approaches related to the traditional family based association study, but focussing on patterns of haplotype sharing, have also been developed and these are discussed in section 1.2.

The main use for genetic association studies is in testing candidate genes and regulatory regions and in fine mapping linkage regions. This means that the short range effect of genetic association can be tested with a dense array of markers. Candidate genes and regulatory regions can be tested for their association by testing whether a particular allele is found to be more common in (usually unrelated) people with the disease than those without. Association mapping can also be used to try and localise a disease causing locus where a linkage region does not contain any obvious functional candidate gene(s) or where there are many plausible functional candidate genes. An association between a phenotype and an allele at a locus indicates either that the allele in question leads to susceptibility to the phenotype or that the allele is in LD with the susceptibility allele. The most basic method of mapping simply involves plotting estimates of association with disease for each marker. The location of the disease gene is estimated to be near the marker with the strongest evidence of association. For an example of association mapping applied, see Corder et al. (1993) who used this approach to show an association of ApoE-4 with late-onset Alzheimer's disease or Davies et al. (1994) for a review of how this approach was used to show an association of a variation in the major histocompatibility HLA region and the insulin gene region with Type 1 diabetes.

The candidate gene approach is the most common strategy for going from a linked region to a gene. There will often be loci of known function that have previously been identified and cloned from the region to which the linkage signal maps. Any of these loci that could potentially give rise to the phenotype linked to the region are referred to as candidate loci. These candidates are then searched for associations with the phenotype. Of course, the candidate gene approach can be used directly, bypassing the need of a linkage study.

To return to the methods for testing association study, as described above, they traditionally break down into two groups: population based, case-control studies or family based studies of the form of the TDT. Population based studies have been more popular with the main reason being the higher statistical power (Morton & Collins, 1998). In addition to reduced power, resources to collect family data are usually higher (in terms of both time and money) than individuals from a population. To achieve the same power, one needs the same number of triads as cases in a case control study, which means extra resources in gathering triads rather than individual

cases and control and also additional extra genotyping (as where one case and one control are genotyped in a case-control study, to gain the equivalent power, three members of a parent child triad would have to be genotyped). However, family based studies have the distinct advantage of being less likely to suffer the effects of confounding.

One of the causes of confounding is population stratification. This refers to the systematic difference in allele frequencies between subpopulations and is often due to different ancestry of these subpopulations. Migration, where individuals from one population migrate into another, is one of the most obvious causes of population stratification. After some generations, population stratification will become less due to admixture. Population stratification can lead to high rates of false positives in case control association studies (Knowler et al. 1989; Lander & Schork 1994; Marchini et al 2004). A number of methods have been developed to detect and correct for population structure (Pritchard & Rosenberg 1999; Bacanu et al. 2000; Pritchard et al. 2000). Another cause of confounding is cryptic, or spurious, relatedness. This refers to the case where non-random mating results in a certain subpopulation that are more related to each other compared to the rest of the population. This is thought also to potentially inflate the false positive rate in some case control association studies (Devlin & Roeder 1999). Voight & Pritchard (2005) published a study into the impact of cryptic relatedness on association studies and they showed that 'for well-designed studies in outbred populations, the degree of confounding due to cryptic relatedness will usually be negligible', however, 'studies where there is a sampling bias toward collecting relatives may indeed suffer from excessive rates of false positives'. They also showed that the impact of cryptic relatedness can be a problem where founder populations that had grown rapidly and recently from a small size. These issues relating to population effects are clearly less of a problem for family studies where controls are related to the individuals under study (Spielman et al., 1993). Family based studies also have the ability to infer parent-of-origin effects (genomic imprinting; Weinberg, 1999). Plus, family-based studies can also offer a solution to model building and multiple testing. In summary, case-control studies have proven popular mainly due to the statistical power/resource play off, but the family-based approach is still often used as a complimentary strategy due to its robustness to population stratification.

1.1.3 Whole genome association

In recent years, since the completion of the Human Genome Project and as the costs have decreased, it has become feasible to study association across the entire genome, these are known as whole genome association (WGA) studies. Genome-wide association studies have been proposed as an alternative to linkage studies. Initial studies have proven to be successful in replicating previously identified loci (Pearson & Manolio 2008) at least in some illnesses (not in the case of psychiatric illnesses). However, due to the large number of markers involved there are issues with the statistical rigour of such studies, in particular the risk of false positives due to the large number of statistical tests that are preformed (Hunter and Kraft 2007, Pearson & Manolio 2008). Other complicating issues include the requirement of large sample sizes, possible confounding due to population substructure and genotyping errors, genetic and phenotypic heterogeneity (Pearson & Manolio 2008). The NCI-NHGRI Working Group on Replication in Association Studies (2007) have tried to create a basic criteria for the reliable reporting of genome-wide association studies, however many published studies have not presented enough detail to assess whether they are taking these factors into account.

1.1.4 Complex genetics

It is important to consider more closely the characteristics of complex genetic disorders and how they affect the implementation of the techniques described above. While the genetic contributions to risk are beyond dispute the genetic architecture of liability to such complex disorders is less clear cut. The major prevailing view holds that multiple common variants, each of small effect size, underlie the genetic liability. This common disease, common variant (CDCV) hypothesis (Chakravarti 1999; Weiss & Clark 2002) is compatible with the commonly observed non-linear decay in risk from proband to first and lesser degree relatives, and with the paucity of evidence for loci of major and widespread effect. A number of high density genome-wide association studies have recently been reported (Wellcome Trust Case Control Consortium (WTCCC) 2007, Diabetes Genetics Initiative of Broad Institute of Harvard and MIT et al. 2007, Tomlinson et al. 2007) that provide evidence for common variants that act as risk factors, but they explain only a modest fraction of the estimated variance. The novel variants discovered in each of the studies were characterised by common minor allele frequency (MAF), in both cases and controls (MAF > 0.067), and modest effect sizes (Odds Ratios (ORs) mostly < 1.5). The WTCCC study looked at associations in cases of seven major common diseases, including BPAD. One independent association was found be significant (P value < $5x10^{-7}$) in BPAD and this variant had a MAF 0.282 in controls and 0.248 in cases and the OR was estimates ~2.1. More recently genome wide association studies of BPAD have been carried out, Baum et al. (2008) found no SNPs of large effect and the strongest signal was found within diacylglycerol kinase ($P=1.5 \times 10^{-8}$), Sklar et al. (2008) also reported no SNPs of large effect and reported two strongly significant SNPs, one in myosin5B ($P=1.66 \times 10^{-7}$) and one in tetraspanin-8 ($P=6.11 \times 10^{-7}$). The CDCV hypothesis does not exclude the possibility that there are also multiple rarer variants of greater relative risk, as recently demonstrated by direct mutational analysis in genes involved in the regulation of HDL cholesterol where rare variants were found to contribute significantly to low plasma levels of HDL-C (Cohen et al. 2004).

There are several challenges specifically involved in identifying the genes related to complex disease such as genetic heterogeneity, where one genetic variant may cause the disease in one family, but a variant in another gene (or a different variant in the same gene) may cause the disease in another family; incomplete penetrance, where an

individual may carry a susceptibility gene, but not present the illness; phenocopies, where an individual presents the illness but does not carry a susceptibility gene and epistasis, where two or more genes interact to create a different effect than would be expected if the genes were expressed independently. While there has been limited success in using linkage and positional cloning in common disorders, some of these problems can be overcome by looking at large families and studying rare subsets of a common disorder that is due to an inherited mutation of large effect. This has led to success in aspects of breast cancer (Hall et al. 1990, Miki et al. 1994), colon cancer (Bodmer et al. 1987) and Alzheimer's disease (Slooter and van Duijn 1997). These successes in identifying variants and genes has not been limited to just the rare subsets used to identify them but they have also proved useful in furthering the understanding of the etiology of the general disease. One of the best examples with application to the field of psychiatric genetics is the study of Stefansson et al. (2002), which found NRG1 to be associated with schizophrenia through a combination of linkage and fine mapping in the Icelandic population.

1.1.5 Copy number variation

Another area that has recently shown potential as a cause of disease is de novo copy number variation (CNV). CNV refers to the variation in the number of copies of a particular DNA sequence that an individual carries. Array-based technologies have been developed to allow DNA copy number variation to be studied on a large scale (Feuk et al. 2006). These methods have recently been used to show that CNV is widespread in humans (Sebat et al. 2004, Iafrate et al. 2004) and have since also been used to show evidence for the association between regions of CNV with autism spectrum disorders (Sebat et al. 2007). Lachman et al. (2007) discuss a number of schizophrenia and BPAD candidate genes that are affected by CNVs and show one significant CNV in the GSK3beta locus (P=0.002). CNV analysis may emerge as a significant alternative in the search for genetic risk factors for disease.

1.2 Allele sharing

To reiterate from the previous section, genetic linkage has been used to help identify genes that are linked to disease and although there has been less success in its application to complex illness, it has still provided strong evidence, albeit of relatively large genomic regions, for loci linked to disease. More recently, the focus has shifted to association studies focusing on candidate regions (many identified through linkage studies) or candidate genes (from functional studies). More recently still, has been the possibility of whole genome association studies. Association studies have greater power and resolution than linkage but although association studies have increasingly reported positive results, replication has often not been so forthcoming. This is most likely due to genetic heterogeneity, low statistical power, multiple testing, variability in study design, phenotype definition, statistical modelling and population structure. While these existing strategies have proven invaluable, allele sharing (or haplotype sharing) provides another approach to investigate allelic association that can compliment the existing methods already described and generate additional information in the search for genetic cause of disease.

In the previous section I also discussed family based association studies, this section will now focus on how an allele sharing methods provide different opportunities when analysing family data. I will discuss how such methods can be used and modified to identify regions of allelic association between large families showing linkage to the same loci. The aim is to use date from families with replicated linkage results for a particular genomic location to be 'recycled' in a between-family analysis. Such an approach seeks to use large, well characterised, families that are linked to a common region to test for the existence of a shared haplotype that would possibly hold a common causal mutation.

1.2.1 Allele sharing background

Test of association in family based studies, such as TDT, are the most popular type of family based test. These type of tests focus on comparing transmitted with non-transmitted haplotypes and in the case of the traditional TDT test, this is carried out in a set of parent-child trios where the child is affected. In the basic case, statistical

significance is usually tested by some goodness-of-fit type test such as chi-squared tests or logistic regression. While the traditional TDT is not able to analyse more than one marker at a time, various extensions have been created to allow the use of multiple markers (Clayton & Jones 1999; Dudbridge et al. 2000; Zhao et al. 2000). Other methods have taken a different approach such as the Haplotype Pattern Mining method which looks at haplotype patterns associated with disease and method such as the Haplotype Sharing Statistic (Van der Meulen & te Meerman 1997) and the Maximum Identity Length Contrast statistic (MILC; Bourgain et al. 2000) which search for excess haplotype identity amongst affected individuals. These methods tend to go beyond just looking at parental haplotypes as they are transmitted (or not) to affected offspring, but to include information on the common inheritance of a haplotype between families. The main benefit from these types of test is that they can incorporate information held within a population while retaining robustness to confounding.

Fan & Lange (1998) described how disproportionately large clusters of affected individuals sharing common haplotypes in the region flanking some disease mutations of recent origin would be expected and Jorde (2000) showed how investigation of the different distributions of the transmitted and non-transmitted haplotypes provides strong evidence for a disease mutation within the extended transmitted haplotype. So, where a disease mutation exists, a region of haplotype sharing flanking that locus may also exist. The corollary of this stands equally: where we find haplotypes that are specifically inherited by affected individuals, we expect such a haplotype to contain a disease causing mutation. The size of the flanking region inherited in common by unrelated individuals depends on the age of the mutation and how distantly related the families or individuals are (de la Chapelle & Wright, 1998). So allele sharing methods are identifying excess allele sharing, or unusually high levels of sharing of consecutive alleles or haplotypes above that which would be expected due to linkage disequilibrium (i.e. that which is seen in the control population).

In the haplotype sharing statistic method of Van der Meulen and te Meerman (1997), they studied markers in the transmitted haplotypes of parent offspring trios and then counted the overlap in shared markers between all pairs of haplotypes. The Haplotype Sharing Statistic (HSS) was calculated as the standard deviation of the shared distance between haplotypes and this was compared to 100 random

permutations of marker alleles over haplotypes using non-transmitted haplotypes. The MILC method (Bourgain et al. 2000, 2001, 2002) also scores transmitted haplotypes in parent offspring trios, but this time, they are compared directly with the scores between non-transmitted haplotypes. Statistical significance is calculated by comparing the maximum difference in scores between the transmitted and non-transmitted groups of haplotypes to the distribution of the maximum difference in this score for randomly permuted datasets of transmitted and non-transmitted haplotypes. Other methods have been developed such as the *haplotype-sharing TDT* (HS-TDT) of Zhang et al. (2003) and the *sequential peeling TDT* (SP-TDT) of Yu et al. (2005) that builds on the sequential peeling method developed for case-control studies (Yu et al. 2004). The sequential peeling procedure goes through case haplotypes sequentially deleting those found not likely to belong to larger clusters creating a clearer set of clusters to test. Statistical significance is again tested through permutation procedures. Most cases tend to rely on permutation testing (Van der Meulen & te Meerman 1997; Bourgain et al. 2000), and Allen and Satten (2007) describe the statistical analysis of these methods as having been developed in an ad-hoc manner and that certainly appears to be the case. This is most likely symptomatic of the fact that these methods were developed in order answer to a specific need rather than as a goal in its own right. As the use of these methods is increasing, there seems to be greater thought put into the development of a statistical basis for these methods. Allen and Satten (2007) go on to describe how permutation testing may be invalidated in the situation where haplotypes are reconstructed due to missing data and there are a number of papers, including their own that try to present a statistical framework for these models. Tzeng et al. (2003a, 2003b) tried to describe the statistical properties of such haplotype sharing methods, describing them in quadratic form and including a very complex description of variance in their model. Beckmann et al. (2005) suggest the use of Mantel statistics similar to the HSC method of Qian and Thomas (2001). Mantel statistics (Mantel 1967) are a method of spatial clustering that can be used to test the distance between groups of haplotypes. These methods are favoured as they may be more adept at detecting relationships between clusters where only a subset of cases are correlated (Legendre 2000). This appears to be a novel application of a technique used in other fields. Allen and Satten (2007) themselves try to create a general framework that for haplotype sharing statistics for the analysis of parent

offspring trios and it should be possible to extend such a framework to accommodate large pedigrees.

Tzeng et al. (2003) and more recently Klei & Roeder (2007) have investigated the power of various goodness of fit and haplotype similarity tests through simulation. Tzeng et al. (2003) compared haplotype similarity tests designed for a sample of unrelated cases and controls and found that frequency-based statistics show relatively more power for low frequency alleles and matching-based statistics work well when the disease allele frequencies are relatively common. In that study, the marker allele was assumed to be the causal allele. Klei & Roeder's (2007) simulations explore a more complex set of scenarios. Similarly to Tzeng et al. (2003), they found that frequency-based methods are more powerful when the disease allele is linked to a low frequency allele (they used short tandem repeats (STRs) rather than SNPs) and haplotype sharing statistic showed more power when the disease allele was linked to a common allele. They also found that the matching and frequency-based approaches had low correlation, suggesting that it is often worthwhile to analyze a given data set with both approaches. Klei & Roeder (2007) also investigated the similarity between the marker and causal allele frequencies. They found that the haplotype sharing statistic achieved its peak power when the marker allele frequency was relatively large, regardless of the disease allele frequency. For the goodness of fit based methods, peak power was achieved when the allele frequencies of the disease and marker alleles matched closely. These results were obtained for a fixed level of LD. They also found that, for high frequency marker alleles, haplotype sharing statistics were more powerful in detecting association. For low frequency marker alleles the goodness of fit based statistics are more powerful. Additionally, the relative performance of goodness of fit based statistics and haplotype similarity statistics depended on how closely the disease allele frequency matches the frequency of the linked marker allele. In conclusion, they reported that the haplotype similarity and goodness of fit based statistics exhibited little correlation and as such both approaches could be used together to optimize the chances of finding association between a disease and a marker allele.

1.2.2 An alternative approach to allele sharing

The intention of this thesis was to study a number of families who share a linkage region on chromosome 4p for bipolar affective disorder (these families are described in section 1.3). Linkage analysis had been carried out and a case control association study was underway. What this thesis aimed to do was to investigate whether there was any additional value to be gained from a comparative study of the four large families that generated the linkage signals. This centred on the possibility that there is some shared ancestry between (at least some of) the families, and that some analysis of the haplotypes of these families might be able to reveal such a shared ancestry. In addition, it was hoped that should such a common ancestry exist, that such an analysis might be used to identify a sub-region defined by the shared ancestral haplotype. Allen & Satten (2007) commented that the statistical basis of most existing methods was designed in a something of an ad hoc manner, and this method is not different. It is with this in mind that the following method was proposed.

Based on the studies of Tzeng et al. (2003) and supported by the later work of Allen and Satten (2007) and others, it was decided that a haplotype similarity test would be the best when studying haplotypes linked with complex disorders that are expected to carry common alleles as such methods that incorporate the haplotype level information should be more powerful. A method was therefore required that would extend the development of some of the existing methods described above for the analysis between multiple large families. The methods of Van der Meulen and te Meerman (1997) and Bourgain et al. (2000) were used as the basis for a new method. As described above, both applied their methods to families, though they focussed on large numbers of trios rather than large families, as have most subsequent methods. The case described in this thesis is that of large families where a consistent disease-linked haplotype can be defined with a degree of certainty due to the ability to phase haplotypes and determine the existence of the observable inheritance of a particular haplotype among infected individuals. This provides a much more certain basis for a test of allele sharing than the uncertain transmission and non-transmission of haplotypes in trios. On the other hand, the scenario described in this thesis is that of a small number of families, so there is a trade off between fewer families, but more certainty in the haplotypes being tested. Another limitation of existing methods (such as MILC, HSS, HS-TDT and SP-TDT) is that the statistical test developed to test

entire haplotypes for a significant difference in sharing. This may be appropriate when only a very small region is being genotyped, but when a much larger number of genotyped covering a large region is available, the number of different haplotypes becomes large and this sort of test is less appropriate. Tzeng et al. (2003a) showed how complex the statistics become when trying to model the results of such a contrast. Permutation analysis provides a simple mechanism for testing the significance of a result and, unlike the haplotype dependent test of Bourgain et al. (2000), it can do so on an individual marker basis. Nested permutation analysis can be used to take multiple testing into account and also to test an expanded haplotype identified by the initial permutation analysis. Allen and Satten (2007) suggested that permutation analysis might not be a valid tool to use, as missing haplotypes require that haplotypes must be permuted based on an invalid model, but this should not be an important issue in the scenario in which this method is being developed where haplotypes are well defined by segregation analysis in a relatively large and well characterised family.

This work started with the assumption that there are multiple families that share a locus linked with a disease and that these families share some common mutation in the region linked with the disease, which is due to a common ancestor. Individuals from each family are required to be genotyped across the relevant region and it is expected that a haplotype that is carried by affected individuals will be identified within each family. This haplotype is referred to as the 'disease-linked' haplotype and there will be one for each family included in the study. The remaining haplotypes will be form a set of control haplotypes for each family. Clearly, not every scenario where one might wish to use such a method will meet these assumptions, however, it was felt that it was more appropriate to have a tool that met the specific requirements of this thesis rather than over generalising and not developing something practical. Having Established a set of disease-linked and control haplotypes from a number of families, the next step is to measure how similar the disease-linked haplotypes are and to contrast this with the similarity found between control haplotypes. Measuring similarity amongst a group of haplotypes was carried out using a scoring system similar to that described by Bourgain et al. (2000). This was judged to be an efficient and accurate means of scoring, marker by marker across the region, of how similar a group of haplotypes are. Scores were allocated to each marker based on the extent to

which the haplotype was shared in both directions around it (details of the scoring system are presented in chapter 2). The same system of scoring was used to score similarity amongst control haplotypes. The differential between sharing at each marker between the disease-linked and control groups of haplotypes can be calculated to give an indication of whether there is some sharing of alleles in one group that doesn't appear in the other. This calculation should also take some account of the LD structure in the underlying population as this would be reflected equally in both disease-linked and control groups of haplotypes.

The next step is to calculate whether any differential scoring is significant. This is the main point of divergence from existing methods, e.g. the MILC method looks at the maximum of the contrast between haplotypes groups (in their case transmitted and non-transmitted). This is probably historical, because methods like MILC were designed for studying a small number of markers as a way to provide further evidence for a region. A new method should take the test a step further, to generate a significance value for markers and for blocks of shared makers, thus identifying any region of excess allele sharing within the wider region being tested. This is done through two stages of permutation analysis. The first stage is testing whether the contrast score between the two groups of haplotypes (disease-linked and control) are significant, by permutation. Permutation testing tests the null hypothesis that there is no difference between haplotypes in either group, so a large number of permutations are carried out with random selection of disease-linked and control haplotypes from the combined set of haplotypes. This produces a significance value for each marker and provides some evidence whether some regions contain markers where there is a significant difference in allele sharing scores in the disease-linked group compared with the control haplotypes. Clearly, if there are many markers being tested, there will be a high number of tests being carried out. A secondary permutation analysis intends not only to correct for multiple testing, but also to take account of the size of the region of sharing. This involves comparing the significance of the results following the first permutation analysis with the results of each permuted dataset following a comparison of a randomly permutated dataset with a large number of others. These analyses are described in more detail in the next chapter in this thesis.

It was felt that these analyses of significance of allele sharing scores was the most transparent and straight-forward approach. There are many assumptions and

simplifications, such as the use of a permutation test rather than a more rigorous statistical test, that made that may allow this method to be somewhat simpler than many other published methods, but it was felt that these assumptions were not too unreasonable, given the data under study in this thesis and that such a scenario is likely to be found in the study of other disorders, where there are multiple linkages to a particular region for a particular disease.

Ideally, this thesis would report the results of more than just the analysis based on the new allele sharing method described above, both to validate any significant results, but also to allow a more practical comparison between methods. While a case-control study of the chromosome 4p BPAD linked region was carried out by others (Christoforou et al. 2007) and discussed in this thesis, there has not been alternative family based analysis. Just such analysis was attempted, however, it was found that published methods were not necessarily easily available for use nor were they always easy to use should they be available. Some comparison of the results of the new method described above and an existing published method are described in chapter 4.

1.3 Bipolar affective disorder

Bipolar affective disorder (BPAD) is amongst the leading causes of disability in the world (Lopez & Murray, 1998). The lifetime prevalence for bipolar I is estimated to be around 1% although rates vary widely between 0.1% and 2.5% (e.g. Faravelli C et al. 1990, Weissman et al. 1996, Szádóczky et al. 1998, ten Have et al. 2002, Regeer et al. 2004, Pini et al. 2005). The annual cost of BPAD to society is estimated to be almost £2 billion per year in the UK alone (1999/2000 prices; Das Gupta & Guest, 2002).

BPAD is characterised by severe mood swings from periods of extreme depression to an overly high and irritable mood (happy, excitable, self-confident, but also angry and impatient), usually with normal moods interspersed. Some people can also experience a combination of depression and mania at the same time (known as mixed state). BPAD was formerly known as manic-depressive illness and was considered to be a well-defined disorder and was seen as distinct from dementia praecox (now known as schizophrenia). Later, the term bipolar affective disorder was used to describe manic-depressive illness, as distinct from unipolar depression. To enable consistent diagnosis, the American Psychiatric Association publishes the Diagnostic and Statistical Manual of Mental Disorders, now on its fourth edition (DSM-IV; American Psychiatric Association). DSM-IV lists many sub-types of BPAD; BPII refers to slightly less extreme version of BPAD and BPIII-BPVI reflect other variations of the disorder. Schizoaffective disorder (SAM) is now used to refer to patients where the mood disorder is similar to BPAD sufferers, but whose mental and cognitive functions are altered in such a way as to suggest the presence of schizophrenia. It would appear that BPAD may share an underlying etiology with unipolar major depression and schizophrenia and there is also genetic evidence for this (Berrettini 2003). BPAD is also associated with mental disorders such as alcoholism and substance abuse (Regier et al. 1990).

Despite the high social and economic impact of the illness, the research carried out to date has been unable to reveal the cellular and molecular function of the disease. As a result, diagnosis tends to rely solely on clinical observation, and treatment tends to focus on alleviating the symptoms rather than the cause. Identifying a causative gene could improve the diagnostic procedure by providing physical evidence for the

disorder and may also allow pre-symptomatic diagnosis and the identification of genetically susceptible individuals. Perhaps most importantly, the identification of a causative gene could lead to the development of drug targets to treat the illness.

1.3.1 Genetic evidence for bipolar affective disorder

There has long been evidence for an inherited aetiology for BPAD. Since the 1920's family and twin studies have established a strong genetic contribution to the disorder. Many studies have documented the increase in risk of BPAD in the relatives of probands with BPAD and the lifetime risk to a first-degree relative of a proband with BPAD is thought to be up to 10%, compared with 1% in the general population (Craddock and Jones 1999). Most of these studies have also found higher than expected levels of unipolar disorder in relatives of BPAD probands. There is also evidence to suggest that relatives of probands with BPAD are also more likely to suffer from other psychiatric disorders within the schizophrenia-affective disorder spectrum (Valles et al. 2000, Berrettini 2003). It is likely that this in part reflects the unreliability of diagnosis, but may also reflect the shared underlying etiology within the range of mental disorders (Berrettini 2003).

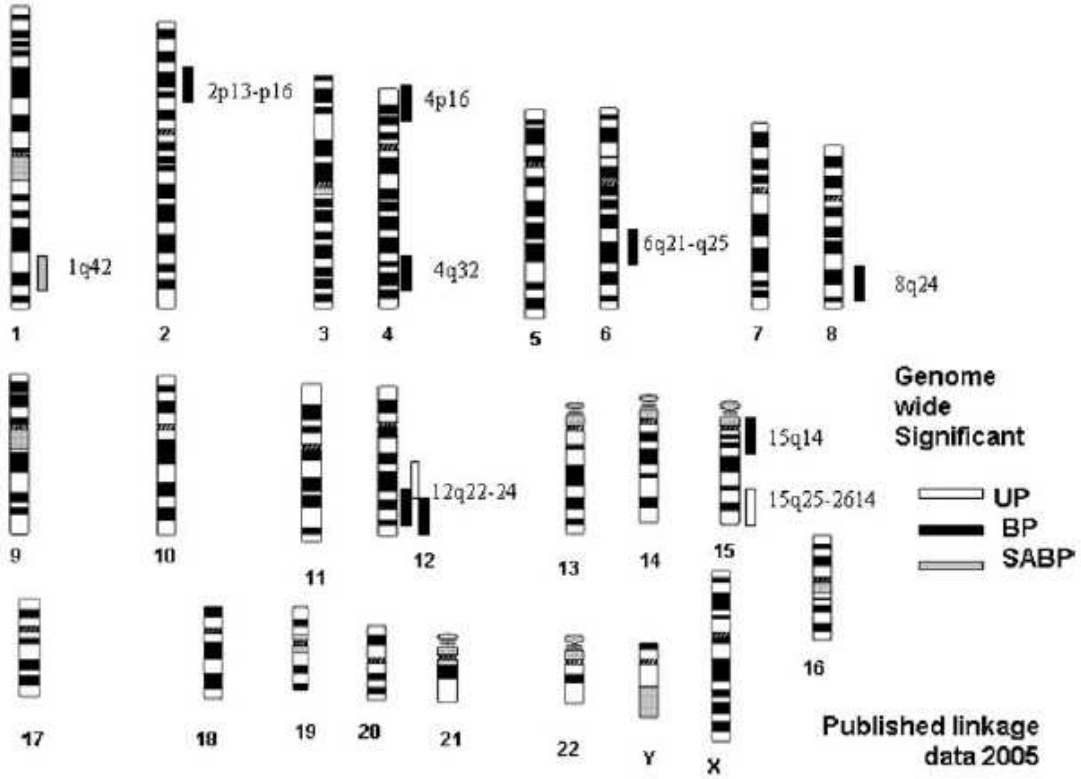
Twin and adoption studies have been carried out to further investigate the genetic cause of the disease. The largest of these used the Danish Twin Register, where they found a proband-wise concordance in monozygotic twins of 0.62, compared to dizygotic twins with concordance of 0.08 (Bertelsen et al. 1977) for BPAD. Craddock et al. (1995) have carried out a survey based on many of the published studies. In combining many published results, they reported a concordance of 0.6 in monozygotic twins, and 0.07 in dizygotic twins and 0.01 in the general population. If common environmental factors can be assumed to be the same for the mono- and dizygotic twins, then under and additive genetic model of gene action, the monozygotic concordance rate should equal twice the dizygotic concordance rate; the large difference between them implies non-additive gene action. The concordance between monozygotic twins is not 1, suggesting that the disorder is not completely genetic in nature, and that environmental elements are also important, which may also interact with genetic risk factors.

While no gene has been shown to directly confer susceptibility to BPAD by positional cloning, a number of linkage and association loci and candidate genes have been

proposed for BPAD and related affective disorders (BP-related). Figure 1.2 presents a summary of significant affective disorder linkage regions published up to 2005. A limited number of these have been widely replicated, but with very few exceptions the evidence for causative mutations is lacking. There are two main exceptions, variants at the D-amino acid oxidase activator (DAOA) locus on chromosome 13q, have been shown to influence susceptibility to BPAD in five different datasets (Hattori et al. 2003, Chen et al. 2004, Schumacher et al. 2004, Williams et al. 2006) and the DAOA gene product is thought to activate the D-amino acid oxidase (DAO) enzyme which is itself found on a region linked to BPAD. *Brain derived neurotrophic factor* (BDNF) is a functional candidate gene with one functional polymorphism that has been reported in three family based association studies to show association with BPAD (Sklar et al. 2002, Neves-Pereira et al. 2002, Geller et al. 2004), although some case control studies have reported no association (Oswald et al. 2004, Skibinska et al. 2004, Hong et al. 2003, Nakata et al. 2003). Other genes such as GRK3 (Barrett et al. 2003), XBP1 (Kakiuchi et al. 2003), P2SX7 (Barden et al. 2006), MAOA (Preisig et al. 2000), COMT (Jones and Craddock 2001) and 5HTT (Anguelova et al. 2003, Lasky-Su et al. 2005) have been implicated but findings remain to be tested. See Craddock and Forty (2006) for a recent review. More recently, Venken et al. (2008) have published evidence of a region on chromosome 10q which displays significant genome-wide linkage with bipolar disorder.

Most cases of family studies of BPAD have comprised sib pairs or multiple simplex families (Levinson et al. 2003). Where large extended pedigrees have been studied, the pattern of disease is usually compatible with a quasi-dominant mode of inheritance, with reduced penetrance (e.g. Blackwood et al. 1996, Morissette et al. 1999, Macgregor et al. 2004 and Herzberg et al. 2006).

Figure 1.2: Genomic regions linked with affective disorders.



Shows the genomic locations that have shown genome-wide significance in at least one scan, for major affective disorder. The predominant phenotype presented in each case is identified as UP: unipolar disorder, BP: bipolar disorder or SABP: schizoaffective disorder (bipolar type). Image reproduced from Craddock & Forty 2006.

1.3.2 Chromosome 4p linkage region

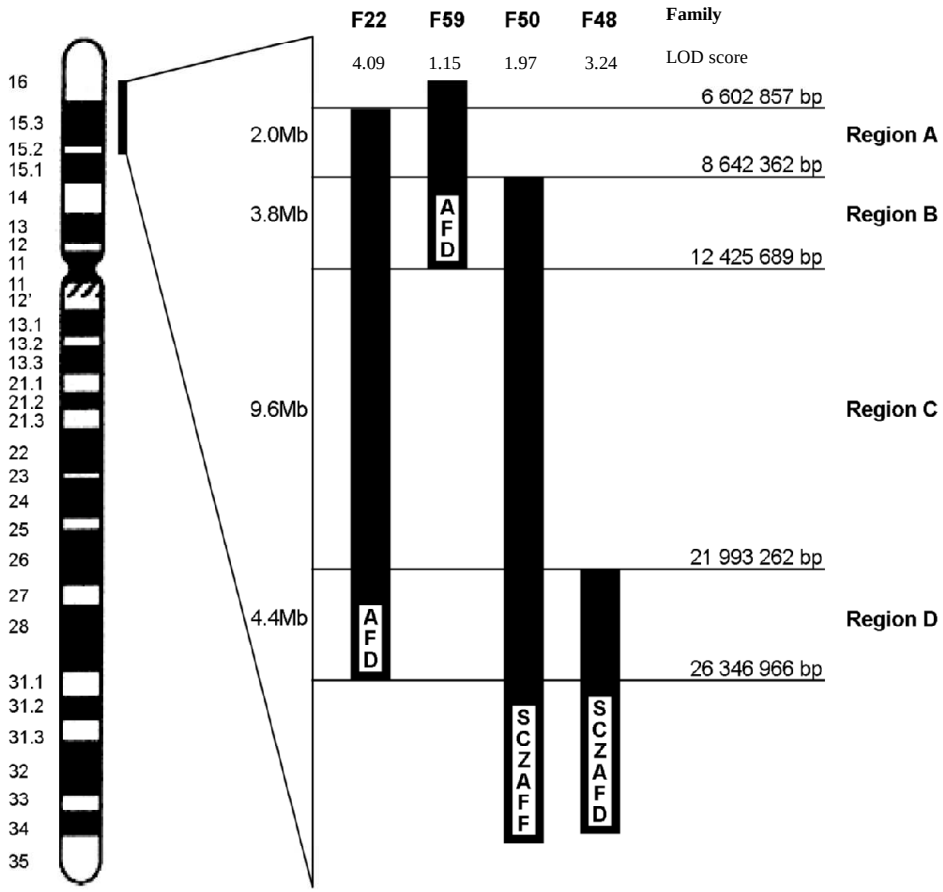
The original evidence of linkage to a region on chromosome 4p came from a genome-wide linkage study in a large pedigree (F22) from the South East of Scotland that segregates for major affective disorder (Blackwood et al., 1996). Patients from the South East region of Scotland has been under clinical investigation by collaborators Prof Douglas Blackwood and Dr Walter Muir (University of Edinburgh, UK) for many years. A whole-genome scan of F22 found significant linkage to chromosome 4p15-4p16, with a LOD score of 4.09 (Blackwood et al. 1996); implying that this regions is likely to contain a susceptibility locus for BPAD. Additional support for this result came from variance component analysis of the same data by Visscher et al. (1999) who found a LOD score of 3.7. Variance component models are based on the

correlation between the genetic similarity of relatives at a given locus and the relatives' similarity with respect to the phenotype. It is very encouraging that an alternative model with differing statistical properties should identify significant linkage at the same region. A re-evaluation of the family was carried out recently (Le Hellard et al 2006) where the clinical status of several family members were updated identifying a maximum LOD score of 4.41 on chromosome 4p16. Evidence from a number of other studies, has since been published in support of this region. In particular, three additional families presented linkage to approximately the same region of chromosome 4p (Blackwood et al. 1996, Asherson et al. 1998, Detera-Wadleigh et al. 1999). Figure 1.3 shows how the F22 linkage region, and of those from three other families, overlap to form two priority regions (*B* and *D*). These families are described in greater detail in chapter 4.

One candidate gene in the region (GPR78) has shown some evidence for association in females with BPAD however sequence analysis of the coding region of the gene has produced no evidence for segregation of the functional variants in F22 or the other three linked families. A recent association study of the region (Christoforou et al. 2007) found three clusters of SNPs and haplotypes that were significant in region B and eight clusters of significant haplotypes in region D. Five known genes were found in these associated regions: GPR125, PPARGC1A, CCKAR, KIAA0746 and DKFZp761B107 all of which would make plausible candidate genes. There are other good candidate genes in the region, including DRD5. However, there is still a lack of evidence to point to a particular gene or region.

The chromosome 4p15-16 region provides a good example of the trend in the study of complex disorders. Many linkage signals have been reported, but the follow ups have been mostly unproductive to date. While some propose that genome-wide association studies can provide an alternative solution, it may be the case that where linkage signals overlap, a new technique can be used to generate more information from the families responsible for these linkage results. In doing so, it may be possible to order to narrow down the region under study even further, thus increasing the likelihood of being able to use the full range of tools available, such as building clone contigs and sequencing the region, as they become cost effective on that scale.

Figure 1.3: Chr. 4p regions of overlap.



*The linked regions that segregate with illness in the four families are found to overlap. Regions A to D indicate sub-regions of the F22 linkage region that show linkage in at least one other family. The sizes (in Mb) of these regions refer to the genomic distances between the points marked by the horizontal lines. The numbers are from NCBI build 35 (*http://www.ncbi.nml.nih.gov*) and are the map co-ordinates of each of the markers that define the boundaries of the linked haplotypes. The illnesses observed in the families are indicated in the figure as follows: AFD – major affective disorder, SCZAFF – schizoaffective disorder and schizophrenia, SCZAFD – schizophrenia, major affective disorder and others. Reproduced from Le Hellard et al. (2006).*

1.4 Aims of Study

The principal aim of the study was to test the two overlapping regions of linkage with BPAD on chromosome 4p for significant allele sharing. However towards that end I have also developed and tested the efficacy of the method, offering a significant new approach in the analysis of complex genetic disorders.

1. In chapter 2 I describe a novel method for testing allele sharing between groups of haplotypes.
2. In chapter 3 I have developed a simulation model to be used in testing the efficacy of the allele sharing method.
3. Also in chapter 3, I have analysed the sets of families with known mutations (presumed to be founder) using the allele sharing method.
4. In chapter 4 I have applied this method to the chromosome 4p linkage region and used this method to greatly reduce the region under investigation and significantly progress the study of this region.
5. The study undertaken in chapter 4 also provides a clear example of how the method could be used for any complex disorder.
6. Also in chapter 4, I have analysed data from the chromosome 4p linkage region with a family-based association test.

Chapter 2

Materials and Methods

2.1 A method for measuring allele sharing

In section 1.2, I proposed a method for testing the level of allele sharing where multiple families present a common linkage to a genomic region for a particular trait or disease. In this section I describe the methods that were developed to do this, as well as describing the preparation of data that was required, the means of generating a statistical interpretation of the results and an overview of how these methods were implemented. Although the methods described here attempts to describe the methods in general terms, there were specific issues relating to the families studied that require discussion also.

2.1.1 Data preparation

The starting point of the analysis described in this thesis is of a number of families identified as sharing some genetic region that is linked in some way with a particular disease or trait (e.g. through a linkage study). Furthermore, these families were required to display a particular haplotype that segregates with the disease or trait in question. It is this 'disease-linked' haplotype which can be compared across families to and assess whether there is significant allele sharing. In addition to simulation studies, a number of families were studied in this thesis and these families all required a great deal of study prior to any allele sharing analysis being carried out. In the cystic fibrosis example, members of the Férec lab identified a number of families that carried one of three known mutations, these families were genotyped at a small number of markers, haplotypes were phased by hand (see section 2.2). In the study of the BPAD and BP-related families, four families had been identified through linkage studies as showing a region on chromosome 4p linked with BPAD, members of each family had been genotyped to a high density, by colleagues, and haplotypes had then been constructed using the MERLIN (Abecasis et al. 2002) software package followed by some refinement of haplotypes by hand (see section 2.4). In each of the above cases, once each family had been phased to the greatest degree possible, the haplotypes of the affected individuals were examined by hand and where it was clear that there was one haplotype in common amongst the majority of cases, that haplotype was identified as a disease-linked haplotype. The assumption was made that if there was some common mutation causing the illness amongst these related individuals,

then it would lie on this haplotype. All other unique haplotypes that were carried by the family were defined as control haplotypes. It is important to note that the allele sharing method carried out in this thesis investigated the allele sharing among disease-linked haplotypes between families and not within families. Similarly control haplotypes were compared between families and not within. So this method is not simply selecting a haplotype that is carried by most cases within a family and testing to see whether this haplotype occurs more often than I would expect by chance, it is testing whether this haplotype shares a sub-haplotype in common with haplotypes found to segregate with the same illness in other families. It is assumed that each family carries a distinct identifiable disease-linked haplotype across the region under investigation, so there is no need to compare within haplotypes in such a way within each family. A description of the general basis of this method is outlined in section 1.2.2 and the specific details of the method are described in the sections following this one.

2.1.2 Scoring

The degree of allele sharing was calculated within the groups of *disease-linked* and *control* haplotypes. A pairwise comparison of the alleles at each marker was carried out between all pairs of haplotypes in each of the two groups. For each pair of haplotypes, each marker was assigned a score based on the size (measured by the number of markers) of the region of sharing in which it was found (based on the methods of Van der Meulen and te Meerman 1997; Bourgain et al. 2000). If a marker was shared between the two haplotypes then the number of markers that were shared around it determined the score for that marker (e.g. if three markers were shared then each marker received a score of three. Scores at markers with ambiguous genotypes were weighted based on the ambiguity (e.g. if it is known that a chromosome could carry one of two alleles at a particular marker then that marker would contribute a score of 0.5 to the block of sharing and a score would be generated for that marker accordingly). Missing data does not contribute to the score of a region, but neither does it break up a shared region, so a marker carrying missing data is scored based on the level of sharing in the markers around it.

Figure 2.1: Allele sharing scoring.

A

Haplotype 1	A	A	C	G	C	G	A	G	G	A
Haplotype 2	T	A	C	G	G	C	A	G	C	T
Score	**0**	**3**	**3**	**3**	**0**	**0**	**2**	**2**	**0**	**0**

B

Haplotype 1	C	A	C	G	G	---	G/C	---	A	G	C	?
Haplotype 2	C	A	?	G	G	C	A	G	C	A
Score	**0**	**2**	**2**	**2**	**0**	**2.5**	**2.5**	**2.5**	**0**	**0**

(A) *is an example of the standard scoring system used to measure allele sharing between a pair of haplotypes.* ***(B)*** *shows how missing and ambiguous data are dealt with. The first instance of missing data (? on haplotype 2) is flanked by one marker on either side that is shared by both haplotypes, so all three markers receive a score of 2. The second instance (? on haplotype 1) is not flanked by any shared alleles, so it receives a score of 0. The case of ambiguity (G/C on haplotype 1) has a 50% chance of being a case of sharing with the allele of the other haplotype, so the score for that marker is 2.5 (0.5 +2, as it is in a region of sharing containing two other markers).*

See Figure 2.1 for an example of the different scoring issues described. The scores from each of the pairwise tests were averaged to generate a *length statistic* for each marker for both the *disease-linked* and *control* groups of haplotypes.

An alternative length statistic was developed based on the physical distance encompassed by a shared region but after some investigation, was not used further during this thesis.

2.1.3 Permutation Testing

Upon calculating a score for each marker, it is desirable to attach a degree of significance to each score. To ask the question of whether the variation in scores are expected or whether they are likely to happen by chance. The distributional properties of the length statistic are complex (Tzeng et al. 2003a), so permutation analysis was used to assess the statistical significance of the null hypothesis of no difference in sharing between linked and control haplotypes.

This permutation process involved the randomisation of all the *disease-linked* and *control* haplotypes. These were then reallocated to *disease-linked* and *control* groups,

while retaining the proportion of haplotypes allocated to each group. The allele sharing analysis was then repeated with these redefined (permuted) haplotype groups. This process was generally carried out 9,999 times. A 'standard' permutation test was then carried out whereby the results of the permuted datasets were used to create an exact distribution of possible differences under the null hypothesis. The P value for the length statistic at each marker was calculated as $p=s/10,000$, where s was the number of times the length statistic for the permutated replicates exceeded the length statistic using the actual *disease-linked* haplotypes. Statistical significance was implied when $P \leq 0.05$.

2.1.4 Correcting for multiple testing

It is important to consider the number of tests being carried out. The permutation test described above are not comparing one score calculated in a pairwise comparison between haplotypes, but one score at each marker. So if there are n individual markers being studied, there will be up to n individual test being carried out. One possibility would be to use a Bonferroni correction, however this is considered to be a fairly conservative approach and even more so considering that some of the markers being tested are likely to be in linkage disequilibrium thus should not be considered as independent tests (Becker & Knapp, 2004). Other tests may require some assumption about the distribution of the data. So, as process of nested permutation analysis, where the permutation replicates themselves were tested against another set of permuted datasets, to account for multiple testing.

Another reason for not using a simple permutation test is that in creating a nested permutation test, we could study much more than just the corrected significance of individual markers. Nested permutations provided the opportunity to take into account the LD properties of the region being studied. This makes the assumption that we know the LD structure (see section 2.4.2 for a description of how the LD structure of the chromosome 4p-linked region was defined). As well as generating the permuted datasets, the initial permutation analysis was used as a filter to identify those areas of interest based on the P values generated. For those regions where the initial permutation analysis had identified at least two consecutive markers with a P value of 0.05 or less, the average allele sharing score was calculated along with the number of LD blocks that were contained within the defined region. The real dataset

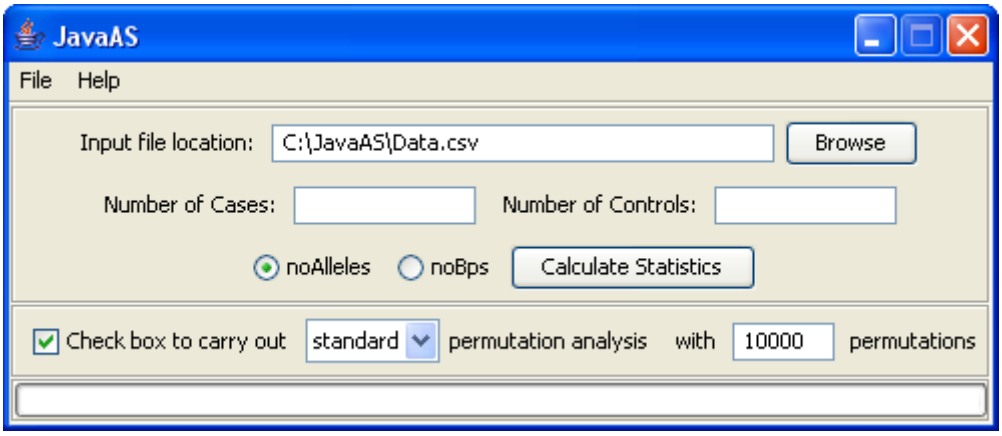
and each permutated dataset, were then compared with another set of 9,999 permuted datasets contained a region of as many LB blocks of the same or greater significance. In a slight modification of a standard permutation test, a P value was then calculated for each region found to be significant by the initial permutation analysis, by P = r/10,000, where r was the number of times a permutation replicate upon its permutation analysis was found to generate a region containing at least as many haplotype blocks as the region found in the linked haplotypes and at a greater or equal level of significance.

Due to computational limitations, rather than creating a further set of permutations to test against each of the original permutations, the remain 9,998 permutated replicates (plus the real dataset) were used in an approximation of nested permutations (Ge et al. 2003).

2.1.5 Implementation

The methods described above were implemented using programs written in Java (Java Platform, Standard Edition, Version 1.4.1; Sun Microsystems). Allele sharing scores and the initial permutation analysis were run through a program called `JavaAS`. `JavaAS` required that the path was specified to a comma delimited text file containing the alleles of the linked haplotypes followed by the alleles of the control haplotypes at each of the markers included in that particular analysis.

Figure 2.2: The `JavaAS` GUI.



The GUI used to carry out allele sharing scoring and permutation analysis.

The program also required that the number of *disease-linked* and *control* haplotypes be specified, the type of length statistic to be used (the length statistic could be based on the number of alleles, or number of base pairs). Finally, the program requires the user to select whether permutation testing is required, selecting the type of permutation analysis and the number of permutations. The program was run through a user interface (Figure 2.2).

Upon running this program, the principle class used to carry out the allele sharing scoring and permutation analysis was called `AsCalculate.java`. In brief, `AsCalculate.java` reads in the data from the comma delimited data file and calls the `calcAS` method (Figure 2.3), which carries out the comparison and scoring of each pair of haplotypes in the linked and the control groups. These are then averaged and difference between the two groups calculated.

Figure 2.3: The `calcAS` method.

A

```
private Vector calcAS(Vector inputData, int colStart, int colEnd){
     Vector pairCases = new Vector();//create a vector of each pairwise comparison
     //create a loop of number of cases factorial (!)
     for (int i=colStart;i<colEnd;i++){
        //within this loop, carry out the pairwise comparison, and enter into
        //the first row of pairComp vector
        for (int j=i+1;j<colEnd;j++){
        Vector pairTemp = new Vector();//new vector for each pairwise comparison
        pairTemp = asLength((Vector)inputData.get(i),(Vector)inputData.get(j));
        pairCases.add(pairTemp); //add the result of each pairwise comparison to
                                 //the pairComp vector
     }
   }//end of loop around the no of pairwise comparisons
   return pairCases;
}//end of calcAS
```

B

```
private Vector asLength(Vector col1,Vector col2){
  Vector pairTemp = new Vector();
  int n=0;//to count where strings of matches occur
  int m=0;//take into account the number of missing points
  int p=0;//counts the number of matches
  double pscore=0.0;//keep track of the mounting score allocated from
                    //ambiguous matches
  for (int k=0;k<col1.size();k++){
    String st1 = (String)col1.get(k);
    String st2 = (String)col2.get(k);
    if (st1.equals("?") || st2.equals("?")){//data missing in hap1 or hap2
      pairTemp.add(new Double(n+pscore));
      if(n>0)//this means that if there is a ?, it only gets a score if there is
             //a score immediately prior, even if there is one immediately after
        m++;
    } else if(!isInteger(st1) || !isInteger(st2)){//if !integer, then must
                                                  //be ambiguous genotypes
      double match = 0;
      if (!isInteger(st1)){//if the hap1 genotype also ambigous
        String st1s[] = st1.split("_");//get the two options
        if (!isInteger(st2)){//if the hap2 is also ambiguous
          String st2s[] = st2.split("_");//get the hap2 options
          for (int i=0;i<st1s.length;i++){
            for (int j=0;j<st2s.length;j++){
              if (isInteger(st1s[i]) && isInteger(st2s[j])){//check that that
                                                    //part of the ambiguity for
                                                    //either hap is not '?'
                if ( (new Double(st1s[i])).equals(new Double(st2s[j])) )
                  match += 1.0;
              }
            }//end of for st2s.length
          }//end of for st1s.lenght
          match /= 4.0;
        } else {//only hap1 is ambiguous
          for (int i=0;i<st1s.length;i++){
            if (isInteger(st1s[i])){//check that this part of the ambiguity is
                                    //not '?'
              if ( (new Double(st1s[i])).equals(new Double(st2)) )
                match += 1.0;
            }
          }
          match /= 2.0;
        }
      } else {//it must be that only hap2 is ambiguous
        String st2s[] = st2.split("_");//get the hap2 options
        for (int i=0;i<st2s.length;i++){
          if (isInteger(st2s[i])){//check that this part of the ambiguity is not
                                  //'?'
            if ( (new Double(st1)).equals(new Double(st2s[i])) )
```

```
            match += 1.0;
          }
        }
        match /= 2.0;
      }
      if (match > 0.0){
        pscore += match;
        pairTemp.add(new Double(n+pscore));
        for(int l=0;l<(n+p+m);l++){
          pairTemp.set((k-l-1),new Double(n+pscore));
        }
        p++;
      } else if (match == 0.0){//there is no match even from the ambiguous data
        pairTemp.add(new Double(0.0));
        n=0; m=0; p=0; pscore=0.0;
      }
    } else if ( (new Double(st1)).equals(new Double(st2))   ){
      pairTemp.add(new Double(n+pscore+1));
      for(int l=0;l<(n+p+m);l++){
        pairTemp.set((k-l-1),new Double(n+pscore+1));
      }
      n++;
    } else {
      pairTemp.add(new Double(0.0));
      n=0; m=0; p=0; pscore=0.0;
    }
  }//end of loop around the length of the pair of columns
  return pairTemp;
}
```

*The first excerpt of code **(A)** shows the `calcAS` method which calculates the allele sharing scores between each pair of haplotypes. The input is a vector matrix holding all the allelic information and markers for the start and end of the group being calculated (linked or control). `pairCases` is a vector used to hold each pairwise comparison in the group and a loop is created to score each haplotype vector against each other. `asLength` **(B)** runs through each allele in the haplotypes and calculates the score, taking account of missing and ambiguous data before returning a vector holding the score at each marker back into the `calcAS` method.*

If permutation analysis is being carried out on the data, the program uses a random number generator to select random haplotypes from the data and place them into the linked and control groups, whilst keeping the number in each group the same as the original data. The program then calls the `calcAs` method for the randomised data. This is repeated for each of the number of permutations specified initially. A method `permStats` (see Figure 2.4) is then called to calculate the statistical significance of the original results by comparing those results with that of the permutations.

Figure 2.4: The `permStats` method.

A

```
permOut = new Vector[noPerms];
for(int i=0;i<noPerms;i++){
  //get random nos
  Vector permTempIn = new Vector();//temp vector to hold the randomised
                                   //input columns
  Random rand = new Random(System.currentTimeMillis());
  Thread.sleep(10);//to make sure there is a unique seed for the random
```

```
                    //number generator
   Vector randList = new Vector();
   for (int j=0;j<inputData.size();j++){//for each haplotype
     int randNo = getRnd(inputData.size(),rand);
     if (randList.contains(new Integer(randNo))){//if we have already taken that hap
       j--;//repeat loop
     } else {
       permTempIn.add(inputData.get(randNo));//add hap to permTempIn
       randList.add(new Integer(randNo));//add to vector randList so it doesn't
                                         // get selected again
     }//end of get random nos
     //use progCount to keep track of the number of permutations that
     //have beencarried out
     progCount = i;
     //now use permuted dataset as the input for the AS score calculations
     permOut[i] = difference( average(calcAS(permTempIn,0,nocases)),
       average(calcAS(permTempIn,nocases,(nocases+nocontrols))) );
     }
   }//end of permutations
   //now calc permutation statistics
   pStats = permStats(permOut, outputData);
 }
```

B

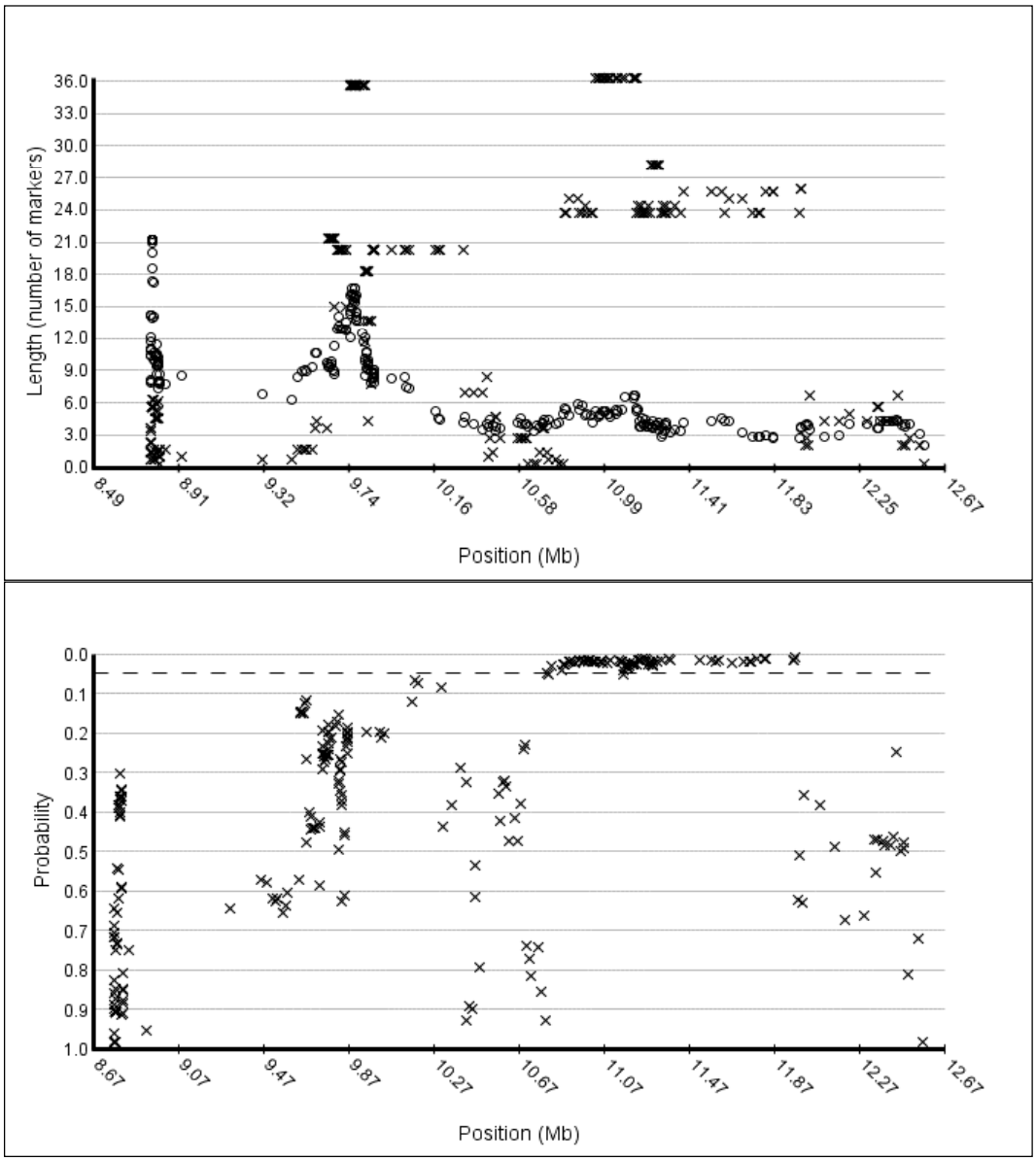
```
private Vector permStats(Vector[] perm, Vector[] out){
   Vector vDiffs = new Vector();
   for (int i=0;i<perm[0].size();i++){
      int temp = 0;
      double nDiffs = 0.0;
      for(int j=0;j<perm.length;j++){
         If(((Double)(perm[j].get(i))).doubleValue()<(
           (Double)(out[2].get(i))).doubleValue())
            nDiffs++;//
         temp = 0;//diff columnn
      }//end of for perm.length (columns)
   nDiffs /= noPerms;
   vDiffs.add(new Double(nDiffs));
   }//end of for perm[0].size (rows)
return vDiffs;
}
```

*The code in part **(A)** is part of the main `AsCalculate` program. Here the program uses a random number generator to select random haplotypes from the full dataset and then call the `calcAS` method described in Figure 2.3. The program loops for n permutations being carried out. The results of each permutation are stored and then used in the call to `permStats`. The `permStats` method **(B)** is used to calculate the statistical significance of allele sharing scores compared with randomly permutated data. This is done be comparing the results of n permutation with the real results.*

A tool to display the results of the allele scoring and permutation analysis was developed. This tool was kept as straight-forward as possible and in the case of the allele sharing scores, simply involved generating a graph that displayed the marker position along an x-axis, allele sharing score along a y-axis. At the position of each marker, a point was plotted for each group of disease-linked and control haplotypes. The two groups were identified through colour and the shape of the point. The graph developed to display the significance value attached to each marker was similar. One point was plotted representing the P value for each marker along a scale on the y-axis. The position of the marker was again along the x-axis. A line representing the 0.05

significance threshold was added and points the lay above the 0.05 value were coloured red. Other features of both graphs were tool-tips that could be used to identify the exact values at any point, a legend and the ability for the user to modify the title of the graphs. Figure 2.5 shows an example of the graphs.

Figure 2.5: ASGraph and PermGraph.



*An examples of the graphs used to display the results of the allele sharing scoring and initial permutation analysis. In **(A)** each marker's position is shown on the x-axis and allele sharing score on the y-axis. At each marker involved in the analysis, two points are plotted,* **o** *represents the score between the control haplotypes and* **x** *the score between the disease-linked haplotypes. **(B)** shows the significance value at each of these same markers. The 0.05 cut-off is shown by the dashed line.*

A separate program was used to carry out the nested permutation analysis that generated the corrected significance values. This was implemented in a program called `NewAsPermAnalysis.java` (some key components of this program can be found in Figure 2.6 below, the full program is listed in Appendix A). The input for this program was the allele sharing score at each marker for the original allele sharing analysis, as well as for each permuted dataset and data on how each marker fitted into the LD structure of the region. Where the initial permutation analysis (described above) compared the real allele sharing score to that found in the permuted datasets, `NewAsPermAnalysis.java` compared the allele sharing scores across a specific number of LD blocks in each of the permuted datasets in turn with the scores for similar sized regions for all the other permuted datasets. This program also took the haplotype structure of the region into account by comparing the average allele sharing score for each haplotype block rather than the score for individual markers. The proportion of permuted datasets that identify regions of greater significance than that found in the original analysis were used to generate a corrected significance value.

The full `AsCalculate` and `NewASPermAnalysis` methods can be found in Appendix A.1 and A.2 respectively. Other programs were created to provide the `JavaAS` GUI and to generate the `ASGraph` and `PermGraph` graphs. These programs are not listed in this thesis.

Figure 2.6: The `NewASPermAnalysis` class.

A

```
Vector realBlockData = getBlockAvs(realData);//get the average score for
                                              //each block, for the real data
//now store the average scores or each block, for each permutation
Vector permBlockData = new Vector();
for (int i=0;i<permData.size();i++){
  permBlockData.add(getBlockAvs((Vector)permData.get(i)));
}
//calc the average of each seq of n blocks for each permutation
Vector permTestNBlocksData = new Vector();
for (int i=0;i<permBlockData.size();i++){
  permTestNBlocksData.add(new Double(getBestNBlockScore(
    (Vector)permBlockData.get(i))));
}

//compare the best score for n consecutive blocks in each of the
//permutations against the real result
int noMoreSigPerms = 0;
for (int i=0;i<permTestNBlocksData.size();i++){
  if(  ((Double)permTestNBlocksData.get(i)).doubleValue() >= realAvScore){
    noMoreSigPerms++;//store the total number of permutations that have a region
                     //of n blocks that are at least as significant as in the real
                     //data
  }
}
```

B

```
private double getBestNBlockScore(Vector blockData){
  //go through each of the list of block averages
  //for te first n-1 add to a vector
  //then for n go on and take the average again
  //then for each additional block remove the first and add the new and take
  //another average
  Vector consecBlocks = new Vector();
  Vector avBlockScores = new Vector();
  double bestNBlockScore = 0.0;
  for (int i=0;i<blockData.size();i++){
    //if we have been through less than the required number of blocks so far
    //add the latest block to the list
    if(consecBlocks.size() < (testNBlocks-1)){
      consecBlocks.add(blockData.get(i));
    }//else if we have just 1 less than required
    else if (consecBlocks.size() == (testNBlocks-1)){
      //add the newest block
      consecBlocks.add(blockData.get(i));
      //and calc the average score
      double sum = 0.0;
      for (int k=0;k<testNBlocks;k++){
        sum += ((Double)consecBlocks.get(k)).doubleValue();
      }
      bestNBlockScore = sum/testNBlocks;
    }//else we then add new block to the end and remove the oldst block from
    //the start
    else {
      //remove the oldest blcock
      consecBlocks.remove(0);
      //add the newest block
      consecBlocks.add(blockData.get(i));
      //and calc the average score
      double sum = 0.0;
      for (int k=0;k<testNBlocks;k++){
        sum += ((Double)consecBlocks.get(k)).doubleValue();
      }
      if ( (sum/testNBlocks) > bestNBlockScore)
        bestNBlockScore = sum/testNBlocks;
    }//end of final else
  }
  return bestNBlockScore;
}
```

*In **(A)**, the average score for each block is calculated and stored for the real data and for the permuted data, the average is also calculated for each sequence of n blocks (where n is the number of blocks found in the region for which the corrected P value is being generated). The `getBestNBlockScore` method **(B)** is then called to calculate the most significant region that covers n blocks in that permuted dataset. The program **(A)** then calculates the proportion of permutations that have a sequence of n blocks with a score higher than the sequence of blocks identified in the real data.*

2.2 Cystic fibrosis study

2.2.1 Cystic fibrosis data

The cystic fibrosis data was based on sixty families from Brittany where some individuals, from each of the families, were known to carry one or other of three cystic fibrosis mutations (W846X2, 9 families; 1078delT, 27 families and G551D, 24 families; De Braekeleer et al. 1996). These families had been genotyped at ten microsatellite markers flanking the CFTR locus with an average spacing of ~900kb and covering a total region of 8.34Mb. Genotyping had been carried out by members of the Férec laboratory (Universite de Bretagne Occidentale) and data from was kindly provided for use in this study.

2.2.2 Cystic fibrosis analysis

Members of the Férec laboratory (Universite de Bretagne Occidentale) had examined the pattern of inheritance in each family studied and a *disease-linked* haplotype common to in the affected individual in each family was defined. A number of *control* haplotypes from each family were also identified. Three datasets were defined consisting of the families that carried each of the three different mutations. Allele sharing scores were calculated for each dataset and 10,000 permutations were carried out on each of these results to test significance at the individual marker level. It was not possible to use the secondary permutation analysis due to the lack of a shared region between the disease-linked haplotypes of different families and so a Bonferroni correction was used to modify the significance threshold to take account of the multiple testing that occurred in the permutation analysis (see section 3.4 for a discussion of this). The modified significance threshold was calculated as $\alpha' = \alpha/n$ where n is the number of tests being conducted. In this case n = 10 as there are 10 markers being tested, so if α is taken to be 0.05, $\alpha' = 0.005$.

2.3 Simulation study

2.3.1 A method for simulating a founder mutation

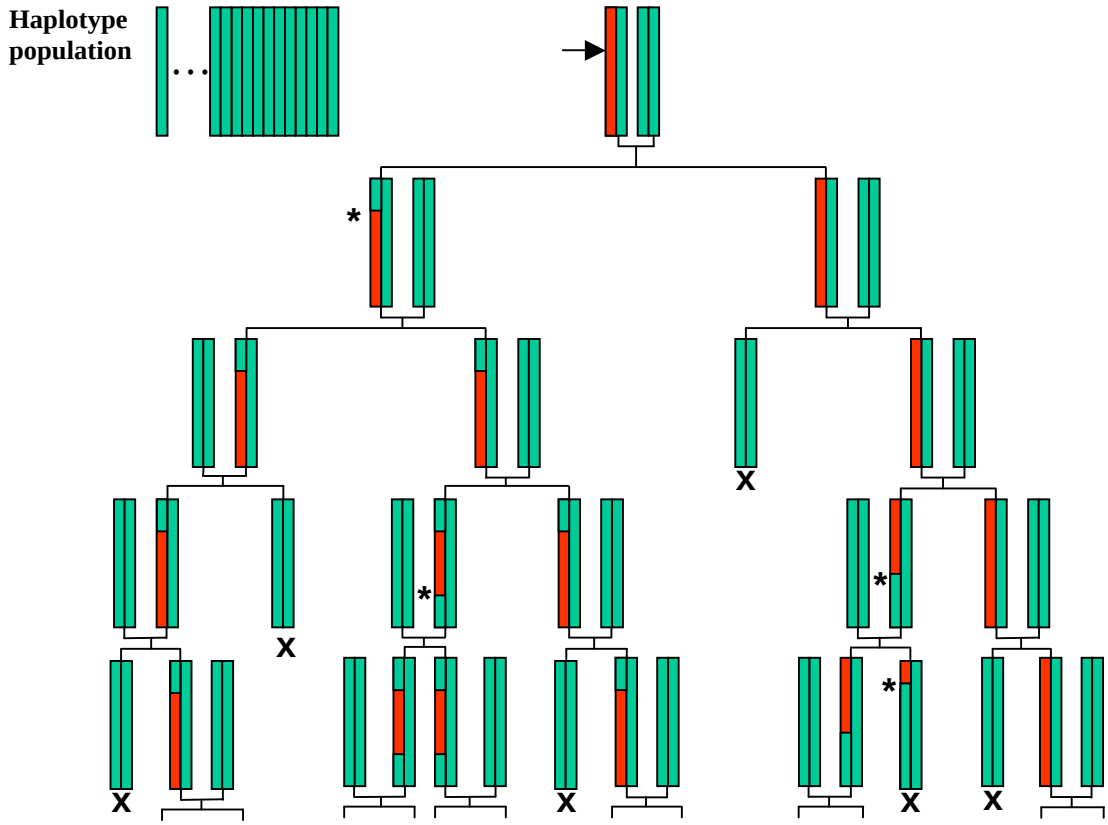
The simulation took a starting population for which the genotypes at a number of markers across a particular region were predicted. A hypothetical mutation at a random point on the genotyped region was recorded. The individuals carrying the mutation were declared to have some phenotype caused by the mutation. The individual who developed the mutation was defined as having had two children with a random member of the population. At each base pair, the chance of a recombination event was defined as $1x10^{-8}$ per generation. These children would each, along with a random member of the population, produce two children themselves and so on. If the mutation established itself in the population, this process would continue for a predetermined number of generations (see Figure 2.7). After which, the most recent generations could be broken up into apparently unrelated families. A pre-specified number of these families would then be chosen on which to test the allele sharing method. One *disease-linked* haplotype and a number of *control* haplotypes from the three most recent generations were identified from each family that was selected. The allele sharing method was then used to identify any regions of significant allele sharing and the locations of any such regions could be compared with the known mutation location to determine success. The simulation makes the assumption that no recurrent, independent mutations occur amongst the control population and that the mutation, and its associated haplotype, can be distinguished from other haplotypes.

2.3.2 Implementation of simulation

The simulation was developed in Java (Java Platform, Standard Edition, Version 1.4.1; Sun Microsystems. Data was simulated through a new program that was developed called `PedSimMain.java`. The program firstly required that the path to a starting population, the number of families that are to be generated and the range into which the numbers of generations between the families and a common ancestor should fall, were all defined on use. The starting population consists of a number of haplotypes each of which contains genotype information at a number of markers on a particular chromosome. The region covered by the genotypes was expected to model

the type of region that a replicated linkage study might identify. There were no set requirements as to the number of haplotypes that make up the starting population and the number of markers that were included in the model other than that each haplotype had to contain genotype information on the same markers. The data was comma delimited with each row representing a marker and each column representing a haplotype.

Figure 2.7: Simulating a founder mutation in a population.



*An example of how the first few generations of a simulation might progress. The arrow represents the random position on the haplotype that the mutation occurs. The haplotype of the original mutation carrier is shown as the red haplotype and all the population haplotypes are shown as green haplotypes. A * indicates where a recombination event has occurred. An X marks where a branch comes to an end as the mutation is no longer carried and is discontinued from the simulation.*

The simulation itself involves an iterative process (for each of n simulations) where the starting population first had one haplotype randomly allocated as the mutation-carrying haplotype with the mutation occurring at a user specified position. A method

`simulator` was used to carry out the core simulation. Figure 2.8 shows the section of the `simulator` method that created each generation of the population and also the `getChildChr` method that models the genetic inheritance in each individual. In order to minimise the computational requirements, only those lineages that carry the mutation were stored and tracked for the entire simulation.

Figure 2.8: `simulator` and `getChildChr` methods

A

```
//for each generation
for (int i=1;i<noGenerations;i++){
  //store the data for each generation in a new hashtable
  peopleHT = new Hashtable();
  //get the data from the previous generation
  prevGenHT = (Hashtable)generationsHT.get((new Integer(i-1)).toString());

  //for each parent from previous generation
  Vector sortedKeys = new Vector();
  sortedKeys = sort(prevGenHT.keys());
  for (int j=0;j<sortedKeys.size();j++){
    //initialise child1
    Vector child0 = new Vector();
    Vector child1 = new Vector();
    //select 50/50 whether to start from parent0 chr 0 or 1
    String key = ((Integer)sortedKeys.get(j)).toString();
    Vector v = new Vector();
    v = (Vector)prevGenHT.get(key);
    //store affection status from 1st parent
    String parentDS = (String)v.get(1);
    //child0
    Vector c0c0temp = new Vector();
    c0c0temp = getChildChr((int[])v.get(2),(int[])v.get(3),(String)v.get(1),key);
    int[] child0Chr0 = (int[])c0c0temp.get(0);
    String child0DS = (String)c0c0temp.get(1);
    //child1
    Vector c1c0temp = new Vector();
    c1c0temp = getChildChr((int[])v.get(2),(int[])v.get(3),(String)v.get(1),key);
    int[] child1Chr0 = (int[])c1c0temp.get(0);
    String child1DS = (String)c1c0temp.get(1);

    //use to select which chromosome and model recombination
    j++;
    String key2 = ((Integer)sortedKeys.get(j)).toString();
    Vector v2 = (Vector)prevGenHT.get(key2);
    //child0
    Vector c0c1temp = new Vector();
    int[] t = (int[])v2.get(2);
    int[] t2 = (int[])v2.get(3);
    String t3 = (String)v2.get(1);
    c0c1temp = getChildChr((int[])v2.get(2),(int[])v2.get(3),(String)v2.get(1),key);
    int[] child0Chr1 = (int[])c0c1temp.get(0);
    //child1
    Vector c1c1temp = new Vector();
    c1c1temp = getChildChr((int[])v2.get(2),(int[])v2.get(3),(String)v2.get(1),key);
    int[] child1Chr1 = (int[])c1c1temp.get(0);

    //add data
    child0.add(key.concat(",").concat(key2));//parentID
    child1.add(key.concat(",").concat(key2));//parentID
    child0.add(child0DS);//disease status
    child1.add(child1DS);//disease status
    child0.add(child0Chr0);//haplotpye1
    child1.add(child1Chr0);//haplotpye1
    child0.add(child0Chr1);//haplotype2
```

```
    child1.add(child1Chr1);//haplotype2

    //if the parent is affected
    if(parentDS.equals("D")){
      //put child1 into the generation
      peopleHT.put(new Integer(idCount++).toString(),child0);
      //if this is not the last generation, make a married in and add
      if(!(i==(noGenerations-1))){
        Vector min = new Vector();
        min = makeMarriedIn(controlHaps[(int)Math.floor(Math.random()*
        (controlHaps.length))],controlHaps[(int)Math.floor(Math.random()*
        (controlHaps.length))]);
        peopleHT.put(new Integer(idCount++).toString(),min);
      }
      //put child2 into the gen
      peopleHT.put(new Integer(idCount++).toString(),child1);
      if(!(i==(noGenerations-1))){
        Vector min = new Vector();
        min = makeMarriedIn(controlHaps[(int)Math.floor(Math.random()*
        (controlHaps.length))],controlHaps[(int)Math.floor(Math.random()*
        (controlHaps.length))]);
        peopleHT.put(new Integer(idCount++).toString(),min);
      }
    }
  }//end of for each person in prev generation

  //now, put this generations info into generationsHT
  if (peopleHT.size()>0){
    generationsHT.put((new Integer(i)).toString(),peopleHT);
  }
  else
    break;
}
```

B

```
public Vector getChildChr(int[] chr0, int[] chr1, String parentDS, String key){
  String ds = "";
  int[] childChr = new int[chr0.length];
  double rand = Math.random();
  int chr2Use = 0;

  //get 1st chr from parent1
  //50% chance of each
  if(rand<=0.5){
    //set chr2Use to 0
    chr2Use = 0;
  } else {
    chr2Use = 1;
  }

  //go through each marker
  for (int k=0;k<chr0.length;k++){
    //add the markers, one by one to the childs chr
    if(chr2Use==0){
      childChr[k] = chr0[k];
      if(k==mutMarker)
        ds=parentDS;
    }else if (chr2Use==1){
      childChr[k] = chr1[k];
      if(k==mutMarker)
        ds="U";
    }
    //test for recombination at each marker
    //get rand no
    rand = Math.random();
    //if rand<(1x10(8)*dbm) then we have a recombination event
    if(rand < (recombRate*distBetweenMarkers)){
      //therefore change parental chromosomes
      if(chr2Use==0)
        chr2Use=1;
      else if(chr2Use==1)
        chr2Use=0;
    }
  }
```

```
  Vector v = new Vector();
  v.add(childChr);
  v.add(ds);
  return v;
}
```

Following the initialisation of variables, ***(A)*** *shows the* `simulator` *method which looped through each generation and modelled the inheritance of the genetic markers.* ***(B)*** *presents the* `getChildChr` *method that was called to model each individual case of inheritance, showing how parental chromosomes were selected randomly and how recombination was taken into account.*

Having modelled the impact of the mutation in a population, the simulator method then proceeded to identify a number of families from the final generations. A family was defined a group of individuals that were directly related in the last four generations of the simulated population. The method then calculates the number of generations separating each of these families. The number of families and the distance between them can then be selected to match the initial criteria determined by the user. Figure 2.9 shows the two sections of the simulator method that select complete families from the most recent generations of the simulation and then identify the most distantly related.

Figure 2.9: `simulator` methods to generate families

A

```
Vector families = new Vector();//store a vector of each family
Enumeration gkeys = generationsHT.keys();//ennumeration of the key(id) for each gen
Vector gkeysSorted = sort(gkeys);//sorted list of these keys
//for fourth last (n-4)th generation, get the hash table
Hashtable ht=(Hashtable)generationsHT.get(((Integer)gkeysSorted.get
 (gkeysSorted.size()-4)).toString());
Enumeration keys=ht.keys();//enumeration of the keys within (n-4)th generation
Vector keysSorted=sort(keys);
//for each individual in (n-4)th generation
for (int i=0;i<keysSorted.size();i++){
  if(((String)((Vector)ht.get(((Integer)keysSorted.get(i)).toString())).get(1)).
   equals("D")){//create a family only if this individual is affected
    Vector family_temp=new Vector();
    Vector individual_temp=(Vector)((Vector)ht.get(((Integer)keysSorted.get(i)).
     toString()).clone();
    individual_temp.add(0,(Integer)keysSorted.get(i));//add key to start of the vector
    family_temp.add(individual_temp);//now add them to the family temp vector
    i++;//get that persons spouse
    individual_temp = (Vector)((Vector)ht.get(((Integer)keysSorted.get(i)).
    toString())).clone();
    individual_temp.add(0,(Integer)keysSorted.get(i));//add key to start of the vector
    family_temp.add(individual_temp);
    families.add(family_temp);
  } else
  i++;
}
for (int i=(gkeysSorted.size()-3);i<gkeysSorted.size();i++){//for 3 subsequent gens
  ht = (Hashtable)generationsHT.get(((Integer)gkeysSorted.get(i)).toString());
```

```
  keys = ht.keys();//enumeration of the keys within generation i
  keysSorted = sort(keys);//these keys sorted
  for (int j=0;j<keysSorted.size();j++){//for each individual j
    String parents=(String)((Vector)ht.get(((Integer)keysSorted.get(j)).toString())).
     get(0);//get j's parents
    String[] parent = parents.split(",");
    for (int k=0;k<families.size();k++){//for each family k
      Vector family = (Vector)families.get(k);
      for (int l=0;l<family.size();l++){//for each member in that family
        //if j has parents in family k,
        if(parent[0].equals(((Integer)((Vector)family.get(l)).get(0)).toString())){
          Integer in = (Integer)keysSorted.get(j);//get individual j's details
          Vector individualj=(Vector)((Vector)ht.get(((Integer)keysSorted.get(j)).
           toString()).clone();
          individualj.add(0,in);//add the key to the start of the vector
          ((Vector)families.get(k)).add(individualj);//add j to that family
          Vector individualjplus1=(Vector)((Vector)ht.get(((Integer)keysSorted.get(
           ++j)).toString())).clone();//also add j++ to that family
          individualjplus1.add(0,(Integer)keysSorted.get(j));
          ((Vector)families.get(k)).add(individualjplus1);
          break;
        }
      }
    }
  }
}
```

B

```
Vector famDistances=new Vector();//vector to hold the comparisons between families
//for the 1st member of each family (apart from the last)
for(int i=0;i<families.size()-1;i++){
  Vector famiid0=(Vector)((Vector)families.get(i)).get(0);
  Vector ijDistances=new Vector();//vector to hold the distances between fami and famj
  for(int j=i+1;j<families.size();j++){// for each other family
    //test this person from famiid0 against famjid0
    Vector famjid0 = (Vector)((Vector)families.get(j)).get(0);
    //initialise parents
    String parents0 = famiid0.get(1).toString();
    String parents1 = famjid0.get(1).toString();
    //store the level of separation
    int sep = 1;
    //go up through all the generations until we find a common ancestor
    boolean comAn = false;//is there a com(mon)An(cestor)?
    while (!comAn){
      //if they have common parents, then store the no of gen separate
      if((parents0).equals(parents1)){
        comAn = true;
      } else {
        //go back a further generation
        sep++;
        //and get the parents from the previous generation
        //generationsHT contains info on all the generations
        int noGens = generationsHT.size();
        //we are starting from the last generation
        //so we now want to get the partners of those individuals named as
        //parents 0 and 1
        //so we look at generation (noGens-sep)
        Hashtable generationN=(Hashtable)generationsHT.get((new Integer(
        (noGens-sep-3))).toString());
        //and find the individuals that are named as parents0 and 1 from
        //need to split up the parents0 and 1
        String[] parent00 = parents0.split(",");
        Vector ind0 = (Vector)generationN.get(parent00[0]);
        String[] parent10 = parents1.split(",");
        Vector ind1 = (Vector)generationN.get(parent10[0]);
        //now, get ind0 and 1s parents
        parents0 = (String)ind0.get(0);
        parents1 = (String)ind1.get(0);
      }//and loop round again with these new parents
    }
    ijDistances.add(new Integer(sep));//store the numb of gens of separation
  }
  famDistances.add(ijDistances);//store the comparisons
}
```

(A) *shows the section of the* `simulator` *method that created family units from the last four generations of the simulation. First it identifies all affected individuals from the fourth last generation of the simulation, it then goes through the subsequent generations and stores together all those individuals that are related.* ***(B)*** *shows how the program then takes these families and calculates the distance (in number of generations) between them by tracing back to a common ancestor.*

Following the identification of a group of families from a simulated population, the program then assumes that the haplotypes from each family can be classified as disease-linked or control. The two haplotype groups from each of the families were then be brought together and used as the input for the `calcAS` (Figure 2.3) and `permStats` (Figure 2.4) methods used to generate allele sharing scores and run the permutation analysis. The haplotypes were then studied (see Figure 2.10 for the code) and shared regions of the disease-linked haplotypes of the simulated families were identified along with information from the allele sharing analysis (`calcAS` and `permStats`). These data were then used as the input for the `NewASPermAnalysis` method which ran the secondary permutation analysis to generate corrected significance values for each shared region. These programs were run automatically without requiring and user interaction. As the program knows where the mutation lies, it took these results and determined whether the allele sharing analysis did indeed identify a significant region that encompassed the simulated mutation. It also determines the number of regions that were found to be significant elsewhere along the haplotype under study (i.e. false positive regions).

Figure 2.10: `FindSigRegions` methods

```
//go through disease-linked haps and search for shared regions
double rSum=0.0;//sum of scores
double pSum=0.0;//sum of p values
int count=0;
String lastBlockNo="";
int numBlocks=0;
Vector blockScore=new Vector();
Vector blockP=new Vector();
Vector blockBlocks = new Vector();
int mutBlock=100;
boolean mutBlockActive = false;
for (int i=0;i<col0.size();i++){//for each marker
  //if all markers are equal store the score and p values
  if(((String)col0.get(i)).equals((String)col1.get(i)) && ((String)col0.get(i)).
    equals((String)col2.get(i))){
     //sum the score and p vlaues
     rSum += (new Double((String)ASScores.get(i))).doubleValue();
     pSum += (new Double((String)ASPValues.get(i))).doubleValue();
     count++;
```

```
    //++ the no of blocks if it is a new block
    if( ((String)blockNos.get(i)).equals("null") )
      numBlocks++;
    else if (  !((String)blockNos.get(i)).equals(lastBlockNo) )
      numBlocks++;
    //if it crosses the mutation, then mark this block as true
    if(i==mutMarkerIndex){
      mutBlockActive = true;
    }
    //if this is the last marker and it is shared, then add as a shared block
    if (i == col0.size() && count > 1  && ((pSum/count) > 0.949)){
      blockScore.add(new Double(rSum/count));
      blockP.add(new Double(pSum/count));
      blockBlocks.add(new Integer(numBlocks));
      if(mutBlockActive){
        mutBlock = blockScore.size() - 1;
        mutBlockActive = false;
      }
    }
  }
  //else if there is no sharing
  else{
    //if previous sharing >1, store the last blocks results
    if(count > 1  && ((pSum/count) > 0.949) ){
      blockScore.add(new Double(rSum/count));
      blockP.add(new Double(pSum/count));
      blockBlocks.add(new Integer(numBlocks));
      if(mutBlockActive){
        mutBlock = blockScore.size() - 1;
        mutBlockActive = false;
      }
    }
    rSum = 0.0;
    pSum = 0.0;
    numBlocks = 0;
    count = 0;
  }
}
//add the last block in it is still being shared
if(count>1  && ((pSum/count)>0.949)){
  blockScore.add(new Double(rSum/count));
  blockP.add(new Double(pSum/count));
  blockBlocks.add(new Integer(numBlocks));
  if(mutBlockActive){
    mutBlock = blockScore.size() - 1;
  }
}
```

Shows how the program goes through the disease-linked haplotype of the simulated families and store the average score and P value within each shared region as well as the number of markers and haplotype blocks covered by that region.

An example of the full simulation program can be found in Appendix A.3. A different version of the program was developed for the different number of families, the example in Appendix A.3 is the program used to create three families.

2.3.3 Initial population and variables

For the purpose of testing the allele sharing method, 458 individuals from the Scottish population were used as the founder population in the simulation. Genotype

information from these individuals at 149 markers over a ~5Mb region of chromosome 4 was used. These individuals constituted the control population of a case control study testing for association with BPAD and/or schizophrenia (Christoforou et al. 2007). This provided a realistic dataset of unrelated, healthy individuals from the same population. The markers genotyped had previously been selected to describe the haplotype structure of the region. On average, there was one marker every ~18kb. For the purposes of the simulation, the simulated mutation was chosen to occur at the 74^{th} marker on the haplotype. Simulations were run on between three and six families and up to 400 generations between the end families and a common ancestor. For computational reasons, it was not possible to run the simulation for more than 400 generations and less than this as the number of families were increased.

2.4 Chromosome 4p data

Four families (F22 F59, F50 and F48) had previously been shown to be linked with BPAD or BP-related disorders at chromosome 4p15-16. These families were studied, in the genomic regions where the linkage signals overlapped, to see if a region of allele sharing could be identified that might signify a common founder mutation.

2.4.1 Families studied

In the original F22 family study, a clinical description was obtained for 120 individual family members. This found 11 with a diagnosed with BPAD, 16 with recurrent unipolar depression and 12 received minor psychiatric diagnoses (Blackwood et al. 1996). Partners were also interviewed to ascertain the bi-lineal descent of affective disorders. This family was recently re-evaluated (Le Hellard et al. 2006), although not all family members could be interviewed. As a result, five new cases were identified and the offspring in one family were removed from the analysis after major psychiatric illness was detected in the first degree relatives of the married in parent. A follow up to the original study (Blackwood et al. 1996) found another family F59, where 11 individuals were studied and six individuals were diagnosed with BPAD. F50 (Asherson et al. 1999) was a family of 16 family members of which five were diagnosed with schizoaffective disorder or schizophrenia. F48 had 39 members that were investigated, six of which were affected under their *affection status model I* (which encompasses in BPAD, major depression and schizoaffective disorder). See figures 4.1 and 4.2 to see the pedigrees of the families and how the linkage signals overlap.

DNA samples from the individuals of the four families were made available. A number of these individuals were selected such that genotypes could be phased to form haplotypes and identify the haplotype common to affected individuals in each family. 31 members of F22, 5 members of F59, 7 members of F50 and 3 members of F48 were genotyped. This data was then subjected to the allele sharing analysis.

2.4.2 Genotyping

Genotyping was conducted in two phases with the first phase based on SNPs found in genic regions within the four families. SNPs were identified by amplifying and sequencing exons, exon-intron boundaries and the region approximately 1kb up and down-stream of known genes from regions B and D using DNA from family members. Known genes were defined as those with predicted, provisional, reviewed or validated RefSeq status codes as well as those that had a protein described in Swissprot. Two genes thought to be unlikely candidates for psychiatric illness met these criteria, yet were not analysed. 216 publicly available SNPs and 68 novel SNPs were identified and used in the analysis (127 in region B and 157 in region D). This work was carried out by colleagues.

The second phase of the analysis was based on a complete coverage of regions B and D. 559 SNPs (175 in region B and 384 in region D) were selected to account for most of the haplotype structure in these two regions. This work was also carried out by colleagues (Christoforou et al. 2007). They downloaded SNP genotype data for the 30 CEPH trios (Utah residents with ancestry from northern and western Europe) (CEU) in overlapping segments of approximately 1 Mb from HapMap Release 7 (May 2004) (*http://www.hapmap.org*). LD maps of the priority regions were constructed from this data using Haploview v 2.5 (*http://www.broad.mit.edu/mpg /haploview/index.php*; Barrett et al. 2005). Pair-wise comparisons of markers more than 500 kb apart were ignored (Haploview default) and only markers with a minor allele frequency (MAF) greater than or equal to 0.10, a Hardy-Weinberg (HW) P-value greater than 0.001 (Haploview default) and a genotyping success rate of 0.75 or better (Haploview default) were included in the LD analysis. Haplotype blocks were defined using the solid spine of LD approach, which creates blocks only when the first and last SNPs are in strong LD (|D'|>0.80) with all of the intermediate SNPs (Barrett et al. 2005). For haplotypes with a frequency of at least 0.01, adjacent haplotype blocks with a Hedrick's multiallelic D' (MAD'; Hedrick et al. 1987) greater than or equal to 0.95 were merged manually and this process was repeated until the MAD' between any two adjacent blocks was less than 0.95. The final step involved, using Haploview's internal tagging program, which selected htSNPs on a block-by-block basis to represent haplotypes of frequencies greater than or equal to 0.10.

Individual SNPs (singletons) that fell between blocks were also included in the set of htSNPs. The LD between them and adjacent blocks was not determined.

2.4.3 Haplotype Analysis

The MERLIN (Multi-Point Engine for Rapid Likelihood Inference) software package (Abecasis et al. 2002) was used to determine distinct haplotypes for the individuals genotyped. The phasing of these genotypes was verified manually and in some cases, where MERLIN had failed to place alleles onto a particular haplotype, this was done by hand. However, in some cases it was not possible to be sure which allele lay on which haplotype. In such cases, this ambiguity was retained in the following analysis. For each family, the haplotypes of the affected individuals were studied by hand and a 'disease-linked' haplotype was identified as that haplotype which was found in common among most affected individuals.

2.4.4 Allele sharing analysis

The allele sharing methods described in section 2.1.4 was used to identify any regions of significant allele sharing between the disease-linked haplotypes.

2.4.5 Association analysis

It was also important that these data were also tested against existing methods, including what could be considered more traditional methods such as TDT. I was unable to get hold of a functioning implementation of any of the published haplotype sharing methods (e.g. MILC or HSS). There are numerous implementations of TDT available through freely available programs such as WHAP (Purcell et al. 2007), FBAT (Laird et al. 2000), TDTae (Gordon et al. 2001, 2004), LAMP ((Li et al, 2005; Li et al, 2006) and other. Each of these programs were investigated to identify what association tests could be carried out. While association analysis at singe marker level was straight forward, there were problems in trying to carry out any haplotype-based analysis. This would appear to be due to computational limitations of the computers I had access to. As a consequence, it was only possible to carry out the most superficial analysis of the data using such methods. The analysis of association

using the TDT method reported in this thesis was carried out using the WHAP program. WHAP is a software tool that performs a haplotype based association analysis described as being designed for both candidate gene studies and studies of small to moderately size chromosomal regions.

The software was used in this thesis to analyse 10 parent offspring trios that were derived from those families (F22, F50 and F59) that showed linkage to region B of the chromosome 4p linkage region for BPAD. The relevant input files were created based on those markers genotyped in region B (see section 2.4.2). WHAP was used to carry out single marker and two-marker haplotype sliding window association tests. 1000 permutations were carried out for each test. One individual haplotype test was carried out on a region found to be significant through the allele sharing analysis.

2.4.6 Gene identification and bioinformatics analysis

Bioinformatics analysis was performed to assess the biological relevance of any significant allele sharing regions. Significant regions were investigated in the UCSC genome browser (Kent et al. 2002) and the Ensembl ContigView (Hubbard et al. 2007) for known genes, protein coding genes based on protein data from UniProt and mRNA data from RefSeq and GenBank; RefSeq genes, known protein coding genes from the NCBI mRNA reference sequence collection; and for main prediction class AceView gene models (Thierry-Mieg & Thierry-Mieg 2006) that do not correspond to Known or RefSeq genes. Evolutionary conservation in non-coding regions in 17 vertebrates was examined using the "Vertebrate Multiz Alignment & Conservation" track on the UCSC Genome Browser. The conservation track is based on a phylogenetic hidden Markov model, phastCons.

Chapter 3

Testing and Understanding the Allele Sharing Method

3.1 Introduction

In the previous chapter, I described a scoring system for calculating allele sharing between a set of haplotypes. I further described how the allele sharing scores of two groups of haplotypes can be compared and tested for significant differences. In this chapter I describe how the efficacy and characteristics of this method were investigated using simulated and real data.

3.1.1 Cystic fibrosis dataset

In order to test the allele sharing method, I required access to a dataset for a number of families that showed linkage to a common region due to a shared founder mutation. Genotype information around the mutation and knowledge of the mutation location were also required. There is, however, a paucity of confirmed founder mutations where a number of families have been densely genotyped. This is because such mutations have usually been found through linkage studies followed by candidate gene studies and it is only recently that it has become cost-effective to genotype a large region to the level at which I expect the allele sharing method to be most effective. However, one appropriate dataset was identified upon which the method could be tested.

This dataset was based on three cystic fibrosis mutations (W846X2, 1078delT and G551D) that had been identified in the Breton population. Cystic fibrosis is a severely debilitating illness that affects the digestive system and lungs and is relatively common in Caucasian populations. The population of Brittany is thought to be a distinct and relatively isolated population where many people are the descendants of the Celtic people displaced from England in the fifth and sixth centuries (Scotet et al. 2002) and the three mutations were found to be much more common in the Breton population than the wider French population (De Braekeleer et al. 1996), it is therefore very possible that these mutations were exist due to some degree of founder effect. The dataset consisted of ten microsatellites subtending the CFTR locus and spanning ~8Mb. This marker density was less than ideal, but I expected that the sparse coverage of the region would be compensated to some degree by the large number of families with each of the mutations.

3.1.2 Simulated dataset

While the cystic fibrosis data provided a reasonable exemplar dataset on which the allele sharing method could be tested, it only provided three mutations with sparse genotyping. Ideally, it would have been possible to go back and genotype a chromosomal region at high density in a number of families known to carry the same founder mutation. However, this was not feasible due to the costs involved. An alternative to using real data was to build a simulation of a population carrying a founder mutation. Simulated data provided a means to test the allele sharing method and had the advantage of being able to generate data under a whole range of conditions. The simulation study involved modelling the development of a mutation in a hypothetical founder population and following its establishment in the population from which, after some number of generations, multiple, seemingly unrelated, families could be gathered to test for allele sharing. This provided a means to generate families under various conditions and more fully test the effectiveness of the method. This also allowed the development of a set of criteria under which the method is expected to perform well.

3.2 Allele Sharing in Cystic Fibrosis Families

A study of Cystic Fibrosis patients in Brittany identified three mutations (W846X2, 1078delT and G551D) that were thought to be prevalent due to a founder effect. 60 families that each carried one of these mutations had been genotyped at ten microsatellite markers about the mutation. This data was used to test the allele sharing method's ability to identify the region that contained the mutation. The data was split into three groups based on the three mutations. For each group of families, a disease-linked haplotype and a number of control haplotypes were defined. These data were then analysed using the allele sharing method.

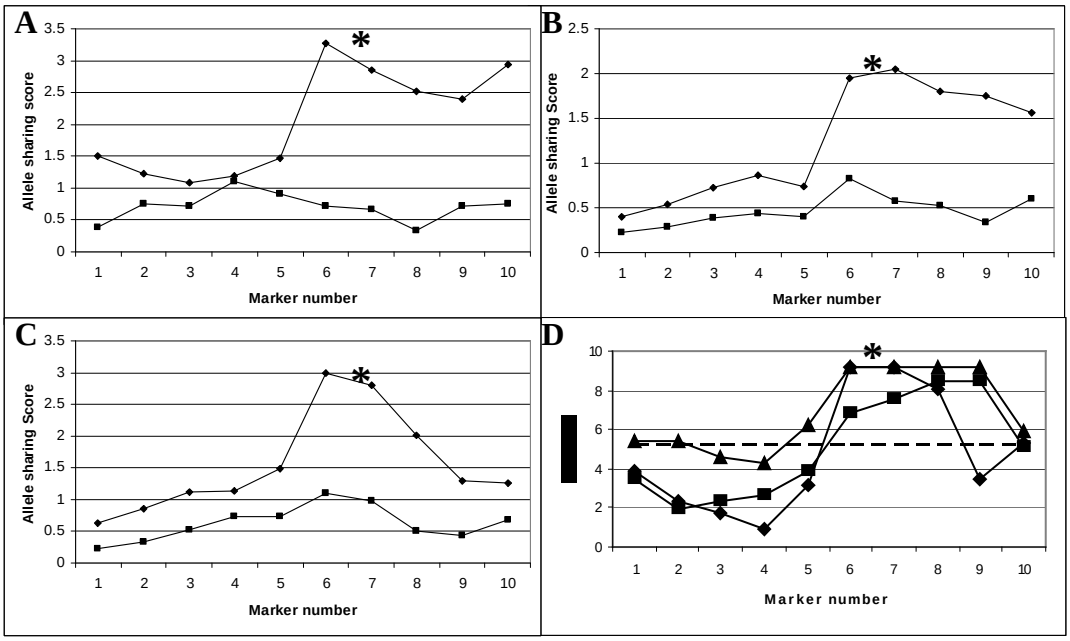
For each dataset (based on the three mutations), allele sharing scores were calculated for both the disease-linked haplotypes and the control haplotypes. Plots of these results are shown in Figures 3.1 (A), (B) and (C). In all three cases, the peak allele sharing among the disease-linked haplotypes occurs at one of the markers immediately adjacent to the mutation. However, in the case of the 1078delT and G551D families, the control haplotypes also show a slight peak in allele sharing at one of the markers immediately adjacent to the mutation, although at much lower levels than for the disease-linked haplotypes. In the W846X2 families, the allele sharing between the control haplotypes is also much lower than the disease-linked haplotypes and in this case, the peak allele sharing occurs three markers away from the mutation location.

Permutation analysis was used to test whether there was a significant difference in the allele sharing between disease-linked and control haplotypes in the three datasets, generating a significance value for each marker. In all of these cases, a significant difference was found between the allele sharing in the mutation carrying and control chromosomes. For the W846X2 families, the markers on either side of the mutation displayed the peak P values ($P = 0.0001$), while the analysis of the G551D carrying families found the peak significance values at the marker immediately upstream of the mutation and the three markers immediately downstream from the mutation ($P = 0.0001$). For the 1078delT families, the peak P value occurred in the two markers adjacent to the marker immediately downstream from the mutation ($P = 0.0002$). Figure 3.1 (D) shows that for the distribution in significance values across the region.

In each of the three datasets, there was no specific haplotype block that was shared by all disease-linked haplotypes. It was therefore not possible to test the likelihood of

these peaks in P values through the nested permutation analysis. The reasons that this might be the case are discussed in section 3.4. As an alternative, a Bonferroni correction can be used to define a modified significance threshold. Based on ten individual tests (one for each of the ten markers) a significance threshold of 0.05 becomes 0.005. In Figure 3.1(D) the significance threshold is marker with a dashed line. It is clear that the mutation carrying region shows significant sharing for each of the three mutations. This is especially apparent for the families carrying the W846X2 and G551D mutations. It is worth noting that the analysis of the W846X2 families also finds markers 1 and 2 to be just significant.

Figure 3.1: Allele sharing in cystic fibrosis families.



(A) *Allele sharing in between disease-linked (♦) and control (■) haplotypes from a number of families known to carry the W846X2 mutation.* ***(B)*** *Allele sharing between families carrying the 1078delT mutation.* ***(C)*** *Allele sharing between families carrying the G551D mutation.* ***(D)*** *The significance value determined for each marker based on the permutation analysis of the allele sharing scores for the three mutations: W846X2 (▲), 1078delT (■) and G551D (♦). The P value is presented as the negative log of the P value. The Bonferroni derived significance threshold is identified as the dashed line.*
*The * indicates the location of the mutation.*

The analysis of each of each dataset is successful in identifying a significant region that contains the founder mutation. In each case, the original region of study of over 8Mb is effectively reduced to 1.53Mb for the W846X2 families, 2.12Mb for the

1078delT families and 3.24Mb for the G551D families. The study of the G551D families also identified a marginally significant region of 2.41Mb that doesn't contain the mutation.

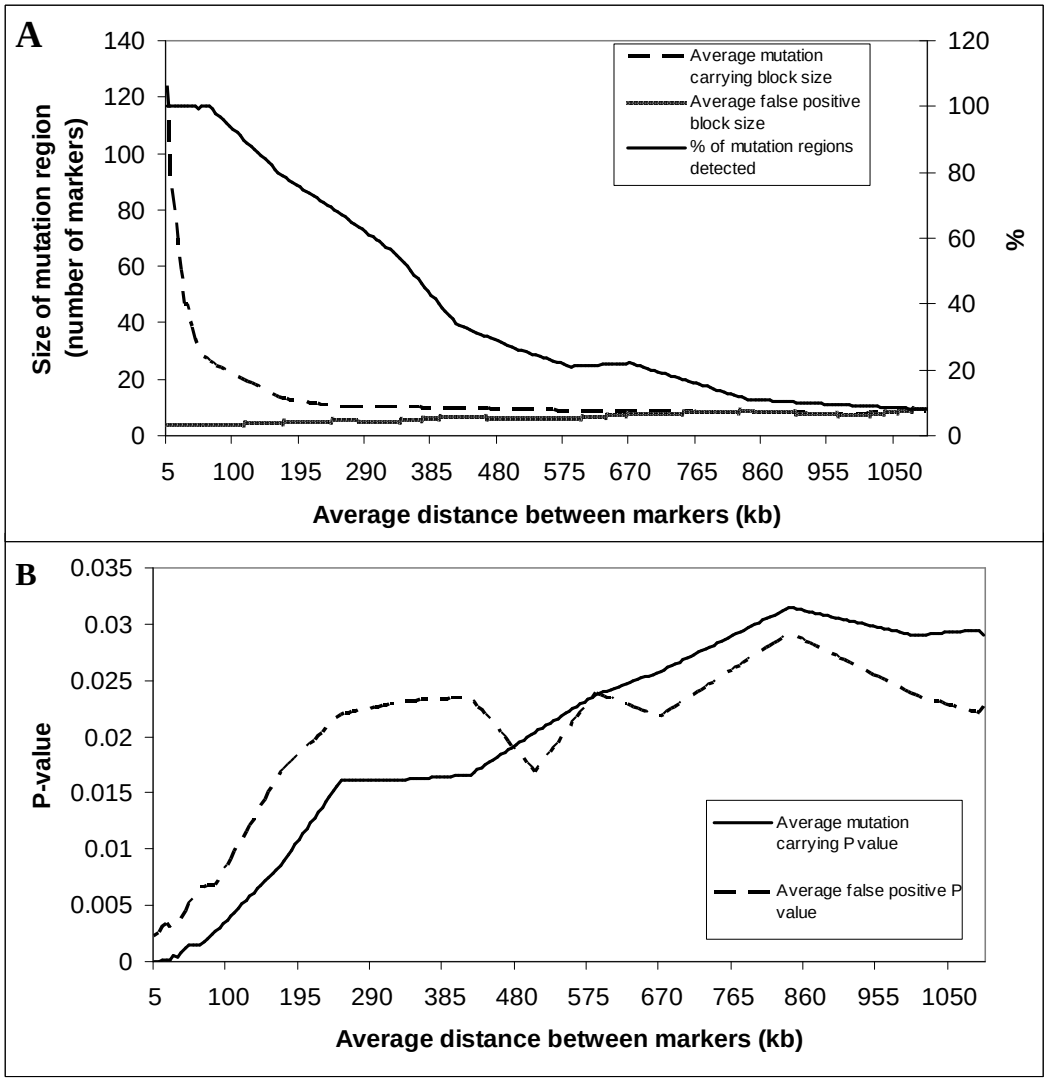
3.3 Defining the operational limits of the allele sharing method through simulation

The simulation was developed to model the establishment of a founder mutation in a population. This allowed us to identify distantly related families carrying the theoretical mutation, providing a number of datasets upon which to test the allele sharing method. There were three main variables in the simulation: the number of families included in the analysis; the number of generations between the families and a common ancestor; and genotyping density across the region. The analysis identified (i) whether the mutation carrying region was detected; (ii) if so, what the size of the shared haplotype region in which it was found was; (iii) what the significance of this result was and (iv) what the size and P values of any false positive regions identified were.

Figure 3.2 displays the results of a series of simulations where the genotype coverage was varied for three families with between 30 to 50 generations (approximately 750-1250 years) to a common ancestor. 500 simulations were run at each datapoint. Figure 3.2 (A) shows that the detection rate remains close to 100% until the genotype coverage drops to 1 marker every ~80kb where after it drops off sharply. Also shown in Figure 3.2 (A) is the average relative size of mutation carrying region (true positive) and the false positive regions. It was found that, as the marker coverage decreases, the relative size of the mutation carrying region decreases and eventually overlaps, and becomes indistinguishable from, the false positives. Figure 3.2 (B) shows that the same pattern occurs for the significance levels (P values), where the significance of the mutation carrying regions falls until it becomes indistinguishable from that of the false positives. The key results were the drop off in the detection rate and the convergence points of the region sizes and P values for the true-positive and false-positive regions. The detection rate falls to the 95% level at marker density of 1 marker per 96kb. At this density, the size and P value associated with the true positives are significantly different.

A similar pattern was observed when the number of families was increased and when the number of generations between the families and a common ancestor was increased, but the convergence points differed. These are discussed in the following section.

Figure 3.2: Sample simulation results.



(A) *Shows the percentage of simulations from which the mutation carrying region is correctly identified for different genotype densities where there are between 30 and 50 generations between three families. Close to 100% of the mutations are found in all cases where the genotype density is greater than one marker per 100kb. Beyond this point, the success rate decreases steadily to a plateau of around 8% success rate when the marker density drops to one marker per 1000kb. The graph also shows that the average size of the mutation carrying region (measured in number of markers) and false positive regions. The size of the mutation carrying regions decrease sharply at a genotype density of one marker per 200kb at which point the size of the region tends towards that of false positive carrying regions.* ***(B)*** *shows that the average P values of the mutation carrying region and false positives steadily increase, linearly and in parallel, up until they flatten out somewhat at a genotype density of around one marker per 240kb. After the marker density drops to about one marker per 390kb, the P values of the two groups converge and then overlap.*

The graphs in Figure 3.2 serve to highlight the minimum genotype coverage required to be confident of finding the mutation carrying region and to be able to distinguish that region from false positives for one particular case. Table 3.1, below, shows the minimum marker density required to give a 95% detection rate for the scenario described in Figure 3.2 as well as a range of mutation ages and number of families being studied.

Table 3.1: Detection limits table.

Age of Mutation		Number of	Expected size of	Min density of markers required to meet
Range	Generations	Families	region (kb)	95% accuracy (kb between markers)
30-50	39	3	1779	116
	40	4	1801	117
	39	5	2458	201
	40	6	2101	208
50-80	64	3	1298	71
	65	4	1424	92
	65	5	1274	109
	65	6	1503	152
80-110	94	3	906	44
	95	4	936	67
	96	5	1004	76
	95	6	1020	91
110-140	124	3	692	36
	126	4	774	48
	126	5	790	60
	125	6	795	68
140-170	156	3	580	30
	156	4	563	43
	-	5	-	-
	-	6	-	-
170-200	186	3	470	27
	187	4	511	36
	-	5	-	-
	-	6	-	-
200-230	215	3	405	23
	216	4	435	31
	-	5	-	-
	-	6	-	-
230-260	244	3	361	20
	246	4	389	26
	-	5	-	-
	-	6	-	-
260-290	275	3	311	18
	-	4	-	-
	-	5	-	-
	-	6	-	-
290-320	303	3	274	15
	-	4	-	-
	-	5	-	-
	-	6	-	-

Table 3.1: Detection limits table (continued).

Age of Mutation Range	Age of Mutation Generations	Number of Families	Expected size of region (kb)	Min density of markers required to meet 95% accuracy (kb between markers)
320-350	335	3	250	14
	-	4	-	-
	-	5	-	-
	-	6	-	-
350-380	364	3	231	12
	-	4	-	-
	-	5	-	-
	-	6	-	-
380-410	395	3	202	11
	-	4	-	-
	-	5	-	-
	-	6	-	-

Shows, for varying age of a mutation, the minimum marker density genotyped across the region in order to achieve a 95% success rate in the detection of a mutation carrying region.

3.3.1 Defining the criteria for the use of the allele sharing method

In addition to understanding the situations where the method proves to be effective in detecting these simulated mutations, it is important to investigate the details of how the method performs in these different scenarios. In this section, the effect of the key variables of the numbers of families, numbers of generations between a group of families and their common ancestor on the relevant parameters is reported and marker density are studied and the output of the allele sharing method compared. The data that corresponds to that described in the following sections can be found in tables 3.2 and 3.3.

3.3.1.1 Varying the marker density

The effect of changing the marker density was investigated by holding the number of families constant at three and the number of generations between the families and their common ancestor within the range of 90 to 120 generations. It was found that as the density of the markers increased, the detection rate of the mutation carrying region, where the significance cut-off was placed at P=0.05, increased significantly while the size of the mutation carrying region fell slightly. At the same time, the P values associated with the mutation carrying region became more significant, as did

the P values associated with the false positives. The number of false positives, however, rose. These results are summarised and quantitated in Table 3.2 (A) and Figure 3.3. Table 3.3 shows how the number of false positives seen relates to the expected number of false positives. As the marker density is increasing, and the number of false positives are increasing, the expected number of false positives is actually decreasing. It is interesting to note that by modifying the significance threshold so that the observed number of false positives matches the expected number, also leads to a detection rate around the 95%.

3.3.1.2 Varying the number of families

The underlying effect of increasing the number of families was studied by keeping the marker density constant (at an average of 1 marker per 80kb) and the number of generations between the families and their common ancestor within the range of 90 to 120 generations. The average size of the mutation carrying region fell slightly as the number of families was increased (from 995kb for three families to 919kb for six families). The P values associated with the mutation carrying region became more significant, however, a similar pattern was found to occur in the P values associated with the false positives. The variation in the number of false positives detected did not correlate with an increasing number of families. These results are also presented in Table 3.2 (B) and Figure 3.3.

As expected, as the number of families included in the analysis was increased, so the level marker coverage required to find the mutation carrying region with a good degree of certainty (≥95%) declined. For example, where the families were separated by between 80 and 110 generations, the required marker density was 1 per 44kb for three families and 1 per 91kb for 6 families (Table 3.1). This had an effect on most of the variables under observation and the combined effect of an increasing number of families with the decreasing marker coverage required to achieve a 95% detection rate was a slight increase in the size of the mutation carrying region detected (779kb for three families to 983kb for six families, in the example where 105 generations separate the families). An additional effect of the decreasing marker density and increasing number of families was a fall in the number of false positives (an average of 2.61 false positives for three families to 1.20 false positives for six families, in the example where 105 generations separate the families). The number of markers

contained in these false positive regions remained constant for different numbers of families (three to four markers), although because the density of markers was falling, the actual (physical) size of the false positive regions rose. Another effect of decreasing the marker density to meet the 95% success-rate point was that the average P value of the mutation carrying region became less significant as the number of families rose (from P=0.0029 for three families to P=0.0042 for six families in our example). A similar pattern was found to occur in the P values associated with the false positives (these decreased from P=0.0095 for three families to P=0.0161 for six families).

Figure 3.3: Detection limits graph.

Shows the drop-off in the genotype density required to detect a founder mutation through the allele sharing method for the increasing age of the mutation for three to six families. Computational limitations meant that the age of the mutation was limited as the number of families rose.

3.3.1.3 Varying the number of generations between the families and a common ancestor

The number of generations between a group of families and a common ancestor was varied between a range of 90 to 120 generations and 340 to 370 generations while keeping the genotype density constant (at an average of 1 marker every 80kb) and the

Table 3.2: Trends in the simulation study results.

Number of families	Marker density (kb^{-1})	Age of mutation (generations)	Detection rate (%)	Size of mutation carrying region (kb)	Size of false positive regions (kb)	Significance of mutation carrying region (P value)	Significance of false positive regions (P value)
(A)							
3	80	105	70.5	995	209	0.009	0.0197
3	60	105	83.3	824	181	0.0078	0.0165
3	40	105	93.8	854	144	0.0043	0.011
3	20	105	98.8	779	128	0.0029	0.0095
(B)							
3	80	105	70.5	995	209	0.009	0.0197
4	80	105	76.1	940	244	0.007	0.0169
5	80	105	81.4	926	241	0.0051	0.016
6	80	105	89.1	919	232	0.0037	0.0149
(C)							
3	20	105	98.8	779	129	0.0029	0.0095
3	20	174	99.3	526	136	0.0016	0.0085
3	20	234	96.9	391	131	0.0044	0.0145
3	20	303	94.1	286	141	0.0061	0.0165
3	20	364	92.6	281	128	0.0076	0.0161

(A) *Shows how increasing the average marker density while keeping the number of families and the average age of the mutation static, leads to a fall in the average size of the mutation carrying region, but an increase in the average number of false positives, and a reduction in the average significance of the mutation carrying region and the false positives. In section* **(B)** *increasing the number of families while keeping the average genotype density and the average age of the mutation static, leads to an increase in the average size of the mutation carrying region, but increase the average number of false positives, and increases in the average significance values for both the mutation carrying region and the false positives.* **(C)** *Shows that if the number of families and the average genotype density are held constant, but the average age of the mutation is increased, then the average size of the mutation carrying region falls, as does the average number of false positives, however the average significance values associated with the mutation carrying region and the false positives also falls.*

significance value were both found to fall as the number of generations was increased. The average number of false positives was found to decrease as the number of generations between the families and a common ancestor increased. The average size of any false positive regions detected was found to remain constant and the average level of significance associated with the false positive regions was found to fall. Again, these results are represented in Table 3.2 (C) and Figure 3.3.

As was found when the number of families rose, as the number of generations between the families and a common ancestor was increased, so the marker density required (for a 95% detection rate) increased (from 1/116kb where three families are separated by 39 generations to 1/11kb where three families are separated by 394 generations; see Table 3.1). Again, the effects of both the increasing number of generations and the increasing marker density need to be taken into account when studying the effect of increasing the number of generations on the 95% detection point. Here it was found that the size of the shared region decreased (from 1779kb to 202kb as the number of generations separating the three families was varied between 39 and 394; Table 3.1). While the number of false positives remains constant (at between two and four cases), the average size of the false positive regions was found to fall (from 449kb when the three families are separated by 39 generations to 40kb for 394 generations). The significance levels remained fairly constant at levels of the 95% detection rate (varying between P=0.0028 and P=0.0045 for the mutation carrying region and between P=0.0097 and P=0.0141 for the false positive regions).

3.3.1.4 Performance of the false positive rate

In order to check on the veracity of these results I studied the false positive rate in the same varying conditions as those described above. These are reported in table 3.3 below which links in to the same scenarios reported in table 3.2 above. It is clear from these results that the observed false positive rate is not acting in the expected manner. Note, that the average expected number of false positives is not 0.05 because there is more than one test being carried out. The number of tests is based on the number of regions of sharing that are identified and therefore this varies depending on how many regions of sharing are found between haplotypes. The ratio of the observed false positive rate to the expected false positive rate varies from the expected ratio of 1 depending on the simulation conditions. This is clearly

problematic as it indicates that the tests used to generate the P values are not accurate. This issue seems to worse affect those cases where there is very dense or very spare genotyping as well as more recent mutations.

Clearly this will have a large impact on any results based on this method and this is something that will be discussed in the next section. However, it is also interesting to view how the significance cut-off can be changed to give the expected level of false positives and what impact this has on the detection rate of the mutation carrying region. These data are also shown in table 3.3. Here we see that those cases where the observed and expected false positive rates are most out of synch, we also see the detection rate vary similarly. So, those cases where we see less false positives than expected, we also see a low detection rate, and those cases where we have a very high level of false positives compared to the expected rate, then we also see a very high detection rate. It is interesting to note that as the significance cut-off is altered to provide an observed false positive rate that matches the expected rate, then the detection rate tends towards the 95%.

Table 3.3: Performance of the false positive rate.

Number of Families	Marker Density (kb^{-1})	Age of Mutation (gens)	Observed false positives (α=0.05)	Expected False positives	P(obs)/ P(exp)	α where P(obs)/ P(exp)=1	Detection rate (α=0.05)	Detection rate (α')
(A)								
3	80	105	0.52	3.64	0.14	0.272	70.5	96.3
3	60	105	1.32	3.49	0.34	0.146	83.3	96.1
3	40	105	2.26	3.26	0.69	0.073	93.3	96.3
3	20	105	4.43	2.59	1.71	0.010	98.8	95.0
(B)								
3	80	105	0.52	3.64	0.14	0.272	70.5	96.3
4	80	105	0.96	2.36	0.41	0.190	76.1	96.5
5	80	105	1.20	1.45	0.83	0.151	81.4	96.8
6	80	105	1.94	2.98	0.65	0.124	89.1	95.2
(C)								
3	20	105	4.43	2.59	1.71	0.010	98.8	95.0
3	20	174	4.01	3.21	1.25	0.059	99.3	95.3
3	20	234	3.11	2.20	1.41	0.061	96.9	96.1
3	20	303	1.84	3.30	0.56	0.072	94.1	94.8
3	20	364	1.52	3.22	0.47	0.078	92.6	95.2

Shows the difference between observed and expected false positive rate for (A) varying marker density, (B) varying number of families and (C) varying number of generations to a common ancestor. Also shows how the significance cut-off could be modified to give the expected false positive rate and what effect this has on the detection rate.

3.4 Discussion

There is a general lack of published datasets from families genotyped at an appropriate density and range about a known founder mutation on which the allele sharing method could be tested. One dataset, however, was forthcoming. This data consisted of three cystic fibrosis mutations that were found in 60 Breton families (De Braekeleer et al. 1996). Individuals from these families had been genotyped at 10 microsatellite markers across an 8Mb region containing the mutations of the CFTR gene. These mutations were found to be much more common in the Breton population than the surrounding French population. Given the relatively isolated nature of the Breton population, it is possible that these three mutations are found to be so prevalent due to a founder effect. If that were the case, we would expect there to be an ancestral haplotype about the mutation that is shared across the families and that this would be detectable using the allele sharing method outlined in Chapter 2.

For each of the three mutations, the allele sharing method found the peak scores between the disease-linked haplotypes to be at one of the markers flanking the mutation, as would be expected if there was a shared ancestral haplotype in the region containing the mutation. In each case, the allele sharing scores was much higher in the disease-linked group of haplotypes than the control group. However, like the disease-linked haplotypes, the 1078delT and G551D families also saw a small peak in allele sharing within the control group of haplotypes at one of the markers flanking the mutation.

Upon testing for significance on an individual marker basis through permutation analysis, it was found that the P values for individual markers peaked at the markers flanking the mutation in two of the datasets (W846X2 and G551D). For the other dataset (1078delT), the peak is displaced by one marker from the nearest mutation flanking marker. This demonstrated that even with the very limited genotyping density around these mutations, the method was correctly able to identify the region containing the mutation. Unfortunately it was not possible to carry out the secondary permutation analysis on these regions as there was no consistently shared disease-linked haplotype between all the affected individuals carrying a particular mutation. This could be due to one or all of the following reasons: (i) the markers used were microsatellites which are relatively mutable and therefore lead to possible differences in alleles on what would otherwise be the same haplotype background; (ii) the

markers are very sparsely spaced, so if the mutation is old enough, the shared haplotype might not be large enough to cover more than one of the markers included in the analysis and (iii) there are a large number of families involved in the analysis, so it is possible that some of these are included due to allelic heterogeneity. However, due to the low number of markers involved and their more sparse distribution, it was feasible to use the Bonferroni correction instead of permutation testing. Using this correction, the region flanking the mutation was found to remain significant for all three mutations.

Based on these results, the analysis of each of these datasets accurately defines a significantly reduced sub-region within which we would correctly expect to find the causative mutation. The study was conducted over 8Mb and for each of the datasets, this is substantially reduced to 1.53Mb, 2.12Mb and 3.24Mb for the W846X2, 1078delT and G551D carrying families respectively. Another region of 2.41Mb was found to be marginally significant in the G551D families. This would appear to be a false positive region, which, given the results of the simulation study, is not wholly unexpected.

A much more extensive test of the allele sharing method was made possible by generating simulated data. Families carrying a founder mutation in a simulated population were generated and used to test the ability of the allele sharing method to detect the mutation carrying region. These families can be though of as apparently independent families that might be identified as showing linkage to a particular region. The minimum genotype density required for there to be confidence in the method successfully identifying the mutation carrying region was defined for a range of numbers of families, and ages of mutations (see Table 3.1.). These results give an indication of the difference in power obtained by varying the number of families versus increasing the genotype density in such studies. For example, if a mutation has been established in a population for 125 generations (~3125 years), three families would have to be genotyped at an average of 1 marker per 36kb, but if a fourth family were to become available, genotyping would only need to be carried out at an average of 1 marker per 48kb. Thus these results can be used to inform study design. They can also be used to estimate the age of a mutation when a mutation has been detected via this method.

The same set of markers were used to represent the haplotypes that formed the population of each of the simulations, regardless of marker density. The difference between the datasets being the distance between these markers. While the increased distance between markers led to an increased breakdown of haplotypes as the generations were simulated, this may have resulted in a higher than usual LD in the starting population due to the markers in the original dataset having been in closer proximity than the simulated data. LD will certainly affect the haplotype transmission through the generations. This may have distorted results at very low densities somewhat. Due to the distribution of the makers, some of the markers were tightly linked together even at low densities. However, as described in section 2.1, this is something that should be accommodated for by the method. At lower densities, even what were the most tightly linked makers used in the study should act independently. Figure 3.2 shows, for one series of simulations, how the changing marker density affects some of the key characteristics. It appears to be the case that at the very lowest marker densities, the method is just picking up random signals.

Due to computational limitations, it was not possible to run the simulation over large numbers of families for very old mutations. Up to 410 generations could be studied for three families, 260 generations for four families and 140 generations for five and six families. However, the trends are made clear by the studies that were conducted. For recent mutations, the method was found to be effective at a low density of genotyping (an average of 1 marker every 40kb would detect a mutation 100 generation (2500 years) old with three families). As the age of the mutation increases so does the density of genotyping required (an average of 1 marker every 11kb would detect a mutation 400 generation (10000 years) old with three families. Every additional family included in the analysis reduces the level of genotyping required.

The simulation was designed to allow any initial population to be used. Rather than generating a completely artificial dataset, it was decided that genotype data from a real control population would be more informative. Although the small starting population is clearly a limiting factor, it was thought that it was more important to use a realistic population within which we could then model the establishment of a founder mutation. It will be useful in future to test the robustness of the simulation to different starting populations. However, although most similar studies use a larger founder population, the starting population of over 450 individuals used here is similar to that used by published simulations (e.g. Bourgain et al. 2000). Another key

approximation of the simulation study was to estimate that each couple would produce two children in each generation. The number of generations was converted into mutation age by estimating that a new generation occurred every 25 years.

The main purpose of the simulation was to test the method in similar circumstances to those seen in the chromosome 4p linked families that are studies in chapter 4. These families were identified as a result of linkage studies identifying linkage to a common region of chromosome 4p. They have been well characterised both phenotypically and genetically. So, although it was clear that the assumption of 100% penetrance and no allelic heterogeneity in the simulation was unrealistic, this did not affect the purpose of the simulation, which was to model how the haplotypes were broken up over the period that was being simulated. Lower penetrance would mean that the affected family members were less likely to display the disorder and allelic heterogeneity would mean that some affected individuals might not carry the same *disease-linked* haplotype. But we are only interested in those families that are likely to identified through linkage or some other means and where we are able to identify a consistent *disease-linked* haplotype for the family. Both these cases are unlikely in a low penetrance, high allelic heterogeneity model. Even in a model with less than 100% penetrance or slight effects of allelic heterogeneity, it is only the ability to identify suitable in the first place and not on the performance of the allele sharing method that will be affected. It was therefore decided that it was an acceptable assumption that no account would be made of these effects in the simulation. Effect size is treated in a similar way to penetrance and allelic heterogeneity in that it is viewed as a factor that will affect the identification of suitable families rather than the analysis of allele sharing between families once identified. It is assumed that although we expect a small effect size of the genetic contribution for psychiatric disorders, the reason families have been identified is because in these cases, we expect to see a mutation of large effect size. Perhaps due to the contributing factors of some other effect. This is exactly the reason why we benefit from looking at families in addition to the population studies. Of course, it would be desirable to carry out a more complex simulation that did account for such factors as this would allow some insight into the processes that produce sets of families such as those under study in this thesis. This was not something that was found to possible within the timescales that this work was carried out in. Although it may be harder to identify suitable families for study where there are confounding factors such as small effect

sizes, this should not effect the ability of this allele sharing method to identify a region of allele sharing should it exist.

The most pressing concern in the results of these simulations studies is that of the number of false positives. It is clear that the number observed in these simulations does not correspond to the expected number. It is likely that there is some feature in the test of significance that is leading to P values being miscalculated. For example, in the case where the marker density of the simulated dataset was increased, the average number of tests was found to decrease. This was expected as we are more likely to find congruous blocks without recombination between markers where the markers are closer together, therefore we would expect to see less tests carried out. With fewer tests, we would expect to see less false positives, yet what we observe is the opposite. When the marker density was as high as a marker every 20kb, the number of false positives matches with the expected number. One explanation for this effect may be that at lower marker densities, P values are being underestimated as the regions of sharing become smaller. This needs to be traced back to the initial method developed however, the results described here also allow us to be aware of this feature and to take it into consideration when analysing any data using this method. So, for lower marker density studies, we can lower our significance threshold and this would increase the likelihood of detecting the mutation carrying region while generating only the expected number of false positives.

Despite these issues, it is apparent from these simulation studies that for reasonable estimates of founder mutation age and effect size, only a modest number of core family datasets, genotyped at affordable marker densities are required to detect allele sharing and thus map founder mutations with high resolution.

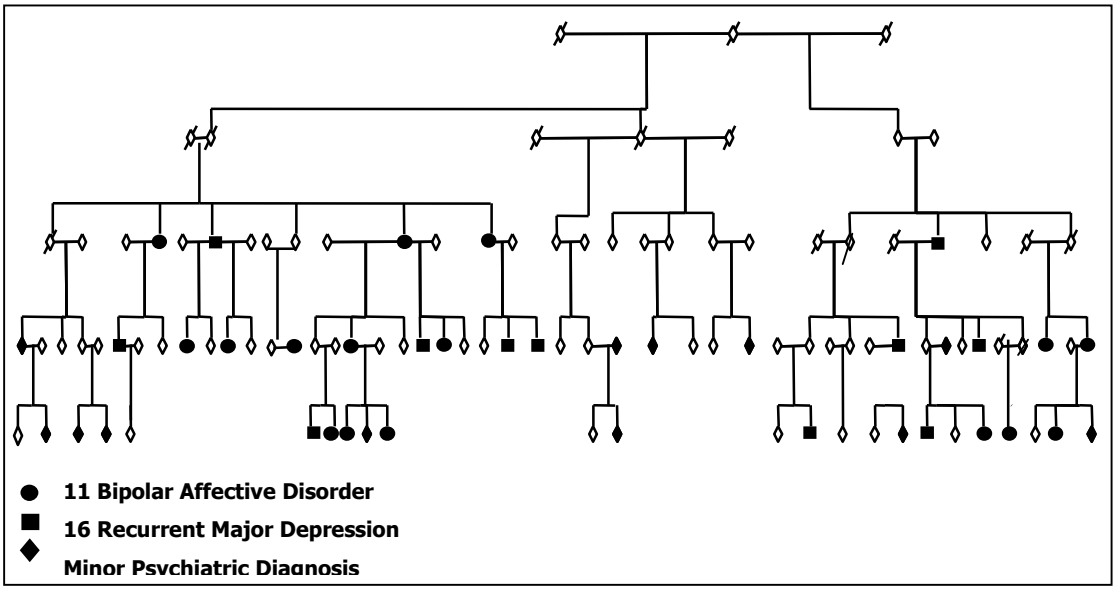
Chapter 4

Allele Sharing in Families Linked with Bipolar Affective Disorder

4.1 Introduction

A significant body of evidence to suggests that chromosome 4p carries a locus of importance for major psychiatric illness. This was first brought to light by Blackwood et al (1996) who described a large Scottish family (F22, Figure 4.1) where many family members presented the symptoms of BPAD or recurrent major depressive disorder (unipolar depression [MIM 125480]). Clinical and genotype data had been obtained from 120 individuals, including 11 with BPAD and 16 with recurrent unipolar depression. A whole genome scan of F22 using 87 microsatellite markers found significant linkage of major affective disorder to chromosome 4p (with a maximum LOD score = 4.09). Variance component analysis of the same data provided further evidence supporting this result (LOD = 3.7; Visscher et al. 1999). Recently, a re-evaluation of the family was carried out (Le Hellard et al, 2006) where the clinical status of several family members were updated. Some family members were newly diagnosed cases and the offspring of one individual was removed from the analysis because major psychiatric illness was detected in a first degree relative of the married in parent. This study identified a maximum LOD score of 4.41 on chromosome 4p16.

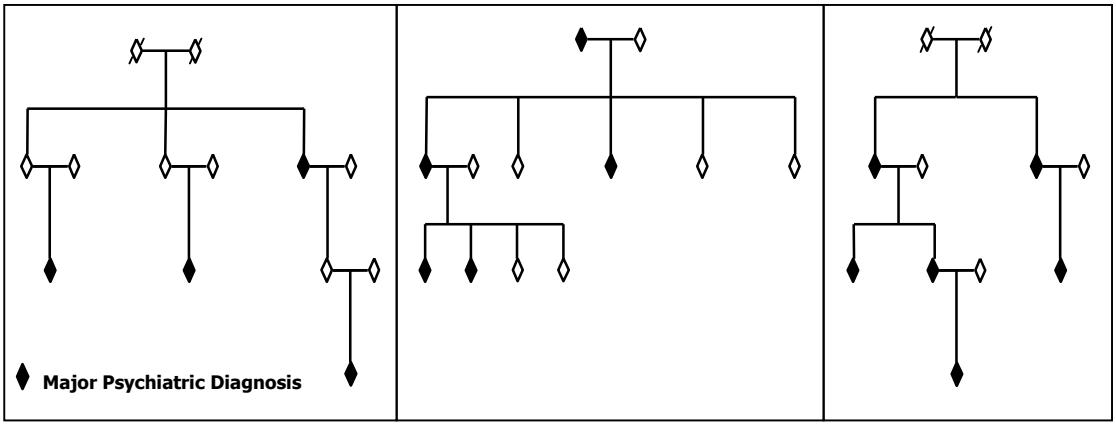
Figure 4.1: Pedigree of Family F22.



Family F22 was the first family that showed linkage to chromosome 4p15-16 for major psychiatric illness.

Blackwood et al. also studied another 56 families and found evidence for linkage in a second Scottish BPAD family (F59; LOD = 1.15; Figure 4.2). This maximum LOD score for this family is limited due to its size, but despite this it still comes very close to meeting the replication criteria as defined by Lander and Kruglyak (1995). Following the Blackwood study, a number of other groups have also reported evidence for linkage to both BPAD and SCZ in this region. Asherson et al. (1998) found linkage in a Welsh schizoaffective family (F50, LOD = 1.97; Figure 4.2.); Ewald et al. (1998) reported linkage in two Danish BPAD families (LOD = 2.00); Detera-Wadleigh et al. (1999) investigated families with major mental illness and their largest family, an American family of Ashkenazi Jewish descent (F48; Figure 4.2), generated a LOD score of 3.24; Williams et al. (1999) found increased sharing in SCZ sibpairs (LOD = 1.73); Lerer et al. (2003) found a non-parametric LOD score of 2.2 in families with SCZ and schizoaffective disorder and Als et al. (2004) found excess haplotype sharing (best P value, P = 0.00007) in families with BPAD and SCZ using an χ^2 test (See Section 1.3 for a discussion of such approaches related to the allele sharing method described in this thesis).

Figure 4.2: Pedigrees of families F48, F50 and F59.

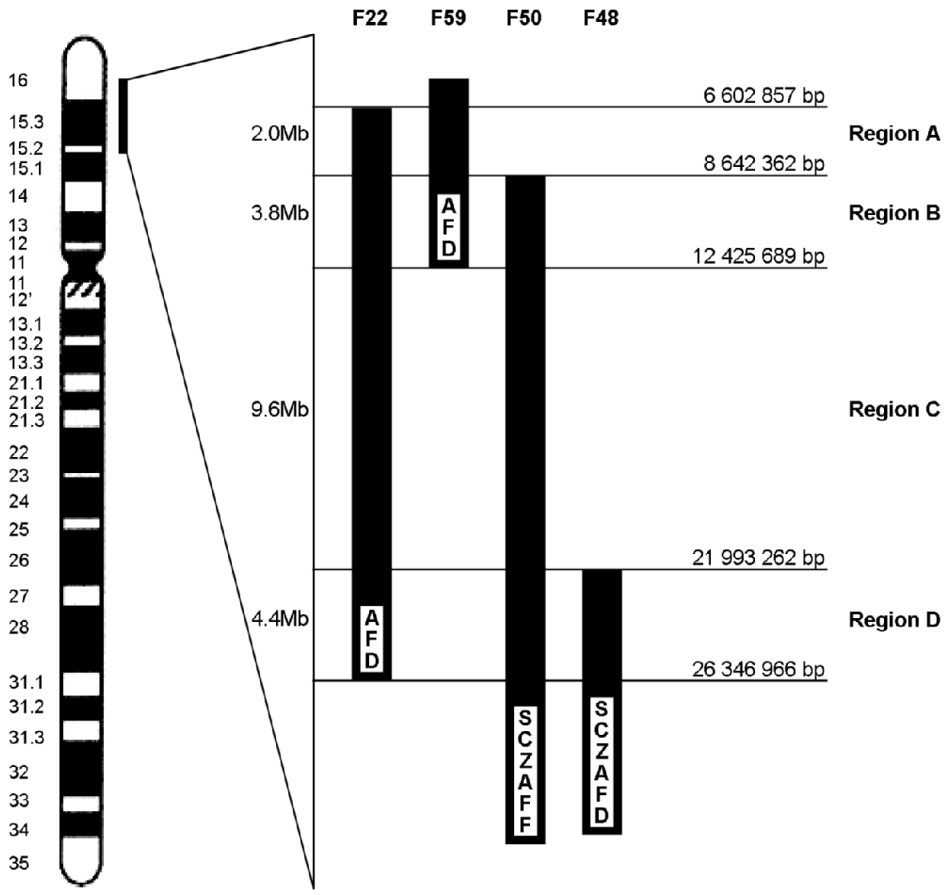


Families (A) Part of F48, (B) F50 and (C) F59 also show linkage to chromosome 4p15-16. Only the branch of F48 that was studied for allele sharing is presented here.

Recently Le Hellard et al. (2006) used a high resolution haplotype analysis of the linked regions of families F22, F59, F50 and F48, to refine these regions of overlap between linkage signals (Figure 4.3). Sub-regions of the F22 linkage signal can be

defined based on the evidence presented. Regions *B* and *D* both have three of the four families showing linkage.

Figure 4.3: Regions of overlap.



*The linked regions that segregate with illness in the four families are found to overlap. Regions A to D indicate sub-regions of the F22 linkage region that show linkage in at least one other family. The sizes (in Mb) of these regions refer to the genomic distances between the points marked by the horizontal lines. The numbers are from NCBI build 35 (*http://www.ncbi.nml.nih.gov*) and are the map co-ordinates of each of the markers that define the boundaries of the linked haplotypes. The illnesses observed in the families are indicated in the figure as follows: AFD – major affective disorder, SCZAFF – schizoaffective disorder and schizophrenia, SCZAFD – schizophrenia, major affective disorder and others. Reproduced from Le Hellard et al. (2006).*

These replicated linkage results could reflect independent mutations at the same locus (allelic heterogeneity) or a common ancient origin (founder mutation). The fact that

there are two priority sub-regions may even indicate the presence of two susceptibility genes (or loci), one in *region B* and the other in *region D*. *Region B* would be the more likely of the two to carry a founder mutation as all three families have Celtic ancestry, while F48 is an American family of Ashkenazi Jewish origin. In this chapter I present the work I have carried out using the allele sharing analysis method to investigate the possibility that founder mutation exists in one of these regions. Initially, most of the known genes in regions *B* and *D* were genotyped at SNPs found within these four families. This was followed up by a study based on the haplotype structure of the entirety of regions *B* and *D*. Markers were selected to describe the haplotype structure of the region. In each of these cases the allele sharing method described in chapter 2 was used to analyse allele sharing based on these data. Here I report those results.

I also report on the basic TDT analysis of region B for the purposes of comparison and validation.

4.2 Allele Sharing in *priority region B*

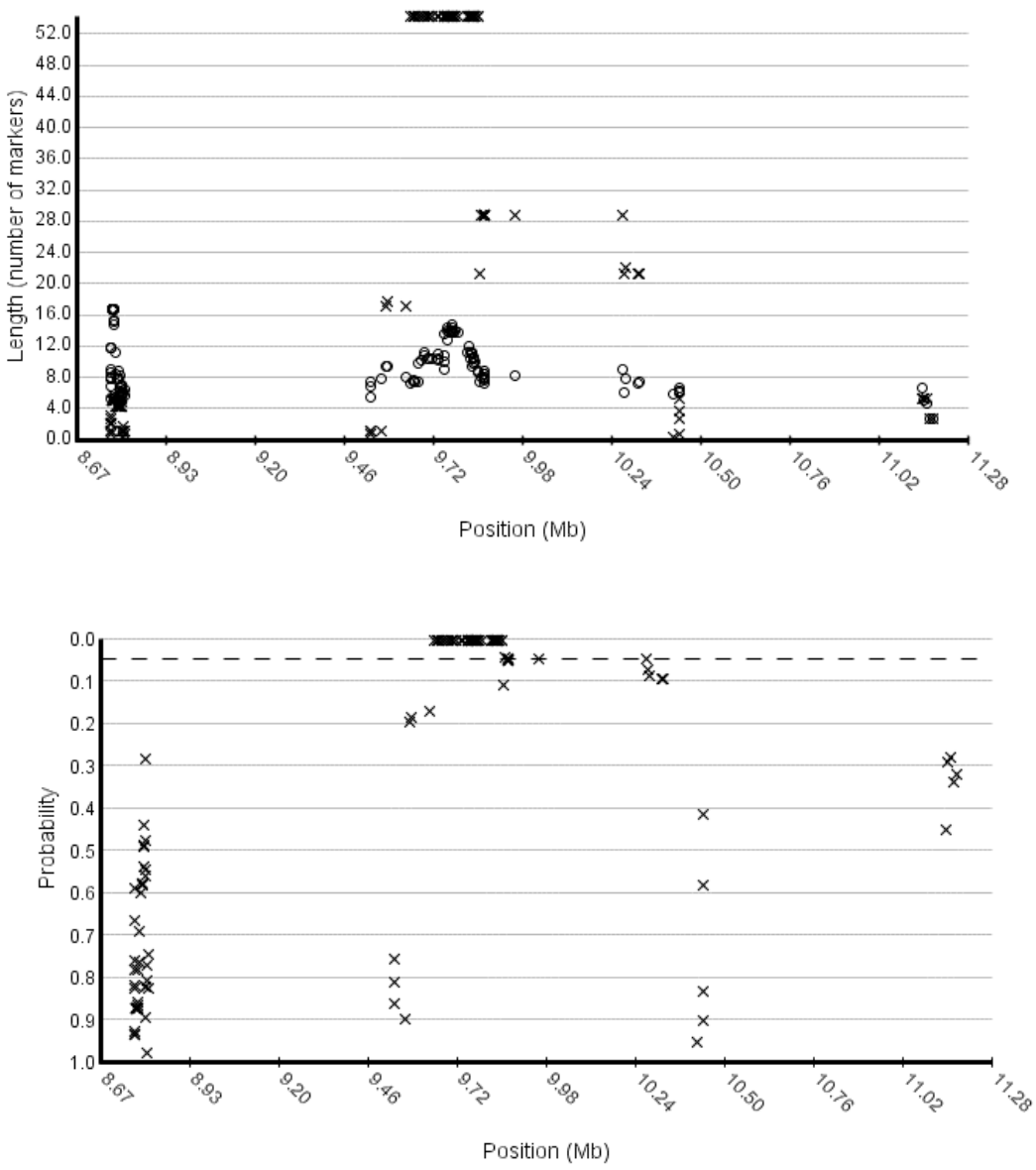
Region B was the first of the two high priority regions investigated. *Region B* is delineated by the overlap of the linkage regions of F22, F59 and F50.

4.2.1 Phase I: Allele sharing at known genes

Initially, markers covering the genes in *region B* were genotyped for reasons explained above. These markers were used to test for excess allele sharing. The results of this analysis are described in Le Hellard et al. (2006). Although individuals from all four families were genotyped, only the three families that show linkage to *region B* are likely to carry an ancestral founder mutation in this region. In this phase of the analysis, three haplotypes (one from each linked family) were allocated as *disease-linked* and 38 haplotypes were determined as *controls* and they were genotyped at 127 markers in the region.

One haplotype consisting of 48 consecutive markers over five LD blocks (covering ~200kb) showed excess allele sharing between the three linked haplotypes when compared to the control haplotypes (see Figure 4.4-A). Permutation analysis of the data indicated that the excess sharing was significant (average P value for the region =0.007, see Figure 4.4-B). The linked haplotype responsible for this result was found in 7.9% of the control samples. This suggests that the *disease-linked* haplotype is fairly common in the population and it could indicate that the *disease-linked* haplotype carries a susceptibility loci rather than a causative mutation. In order to correct for multiple testing, the secondary permutation analysis was carried out that tested how often a region of equivalent or greater significance would occur by chance. A region of sharing across this number of LD blocks and with this level of significance was found in only a small percentage of permuted chromosome sets, giving a corrected P value of P=0.009.

Figure 4.4: Phase I: Allele sharing in *priority region B.*



Shows allele sharing for markers genotyped around genic regions of region B. ***(A)*** *Allele sharing scores and* ***(B)*** *significance values determined through permutation analysis. Reproduced from Le Hellard et al. (2006).*

There are three other groups of markers in *region B* that are significant on an individual marker basis. The first of these consists of nine markers within one LD block; these markers are separated from the significant region described above by just one marker (and 10kb). Here permutation analysis to assess significance gives an

average P value of 0.05. However, when the additional permutation analysis is carried out to correct for multiple testing, this region is no longer significant (P=0.193). Secondly, there is an individual marker with a P value of 0.048 that is corrected to 0.195. The final region is an individual marker that covers one LD block with a P value of 0.049. After correction for multiple testing, this is no longer significant (P = 0.215). A caveat to this analysis of the above three regions is that, unlike the significant region described above, the limits of these regions are not defined by a lack of sharing between the linked haplotypes, but by the end of a gene. The further genotyping that was carried out for *phase II* allowed a more accurate measure of the significance of sharing in these regions.

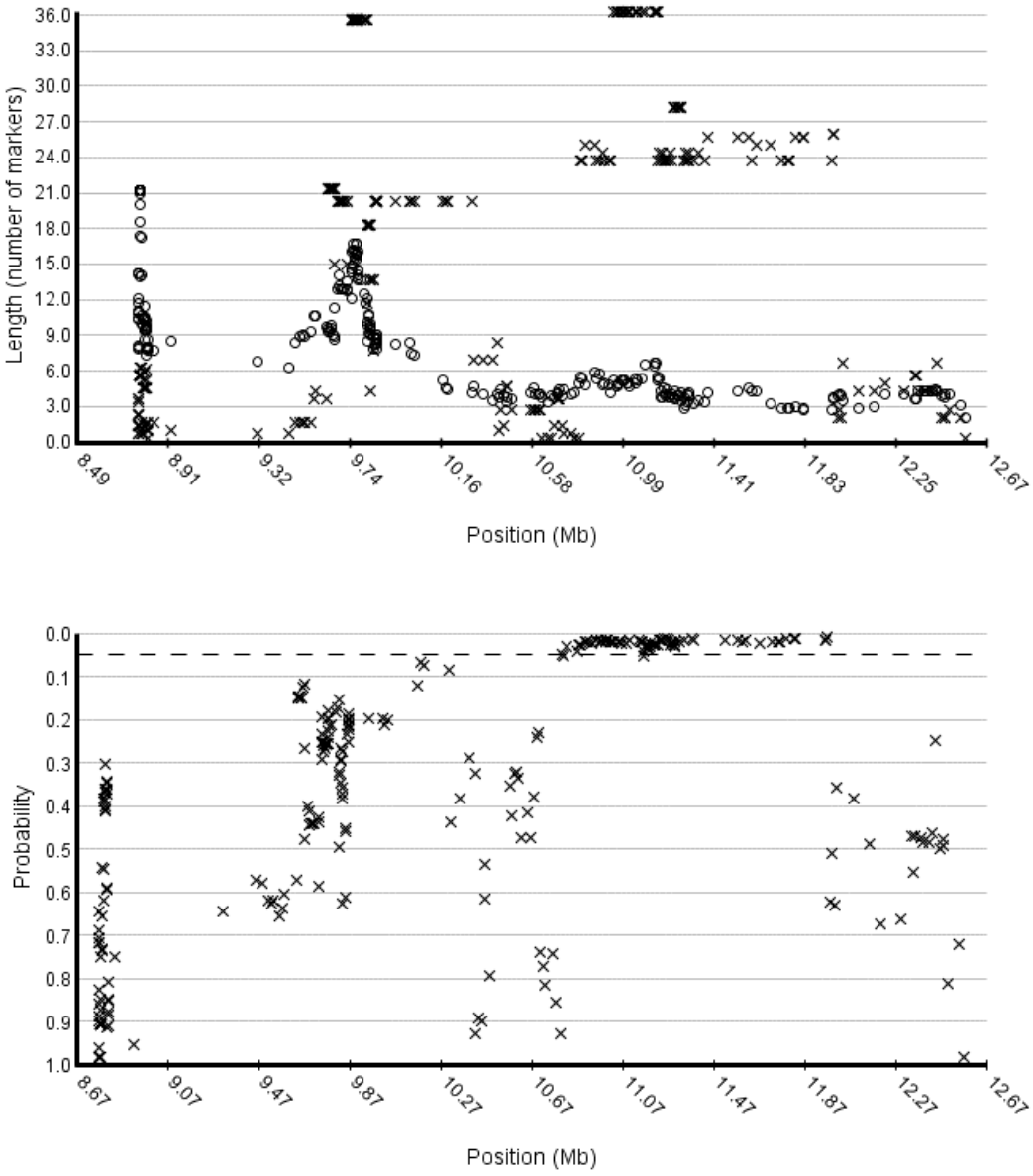
4.2.2 Phase II: Allele sharing across the region using haplotype-tagging markers

Following on from the study of the markers found in genic regions, a study of the entire region was undertaken. 175 markers were genotyped across the region. As before, three *disease-linked* and 38 *control* haplotypes were identified and tested for excess allele sharing.

Figure 4.4(A) shows the allele sharing scores between disease-linked and control haplotypes in *region B* when the haplotype-tagging markers are studied. There are two areas where the sharing between disease-linked haplotypes is noticeably greater than that seen in the controls. The distal region corresponds to a peak in sharing between the control haplotypes. Permutation analysis of this data, Figure 4.4(B), identifies a large significant region that corresponds to the region that displayed much greater allele sharing scores in the disease-linked haplotypes compared with the control haplotypes. There is also a second smaller significant group of markers that appear to be significant, corresponding to a region of high sharing between disease-linked haplotypes, but the significance level falls off where the sharing between control haplotypes reaches a peak. We carried out a multiple testing correction on all the regions where the disease-linked haplotypes are common by descent within these significant regions and at least one marker was significant on an individual marker basis. Table 4.1 shows 11 of these shared regions that were found to be significant after this test was carried out. One of these regions, *shared region 1*, which spans 197kb or 17 LD blocks, is much more significant than the others (P = 0.008). The

rest of the significant regions have an average P value of 0.035 and cover an average of 64kb or 3.67 blocks. *Shared region 1* is found at NCBI Build 36 March 2006 assembly coordinates: 10,814,974 - 11,012,802. The *disease-linked* haplotype that forms shared region 1 is found to be common amongst controls, ~21% of whom carry the haplotype. Again, this may indicate that the mutation is common and only increases susceptibility to BPAD rather than actually causing the disorder.

Figure 4.5: Phase II: Allele sharing in *priority region B*



Allele sharing for markers tagging haplotypes across all of region B. ***(A)*** *Allele sharing scores and* ***(B)*** *significance values derived through permutation analysis.*

Table 4.1. *Priority Region B* shared regions.

Shared region	No. shared markers	No. LD blocks shared	Length of shared region (kb)	Individual P value	Corrected *P* value	Genomic location
1	18	17	197	0.007	0.008	10,814,974 – 11,012,802
2	1	1	n/a	0.017	0.046	11,023,135
3	1	1	n/a	0.009	0.048	11,034,055
4	1	1	n/a	0.012	0.05	11,071,052
5	7	6	38	0.005	0.026	11,088,565 – 11,126,790
6	1	1	n/a	0.013	0.046	11,144,685
7	1	1	n/a	0.006	0.048	11,200,245
8	3	3	191	0.014	0.041	11,243,225 – 11,435,085
9	2	1	62	0.014	0.046	11,468,817 – 11,531,810
10	3	1	41	0.012	0.036	11,647,264 – 11,688,697
11	2	2	3	0.01	0.039	11,821,735 – 11,824,876

The table lists the properties of the regions of excess sharing in region B of the F22 linkage region, the number of shared markers within them, the number of LD blocks they cover, their length, the P-value after the test for significance and the P-value corrected for multiple testing and their genomic location (chromosome 4 NCBI Build 36 March 2006 assembly coordinates).

4.3 Allele Sharing in *Priority Region D*

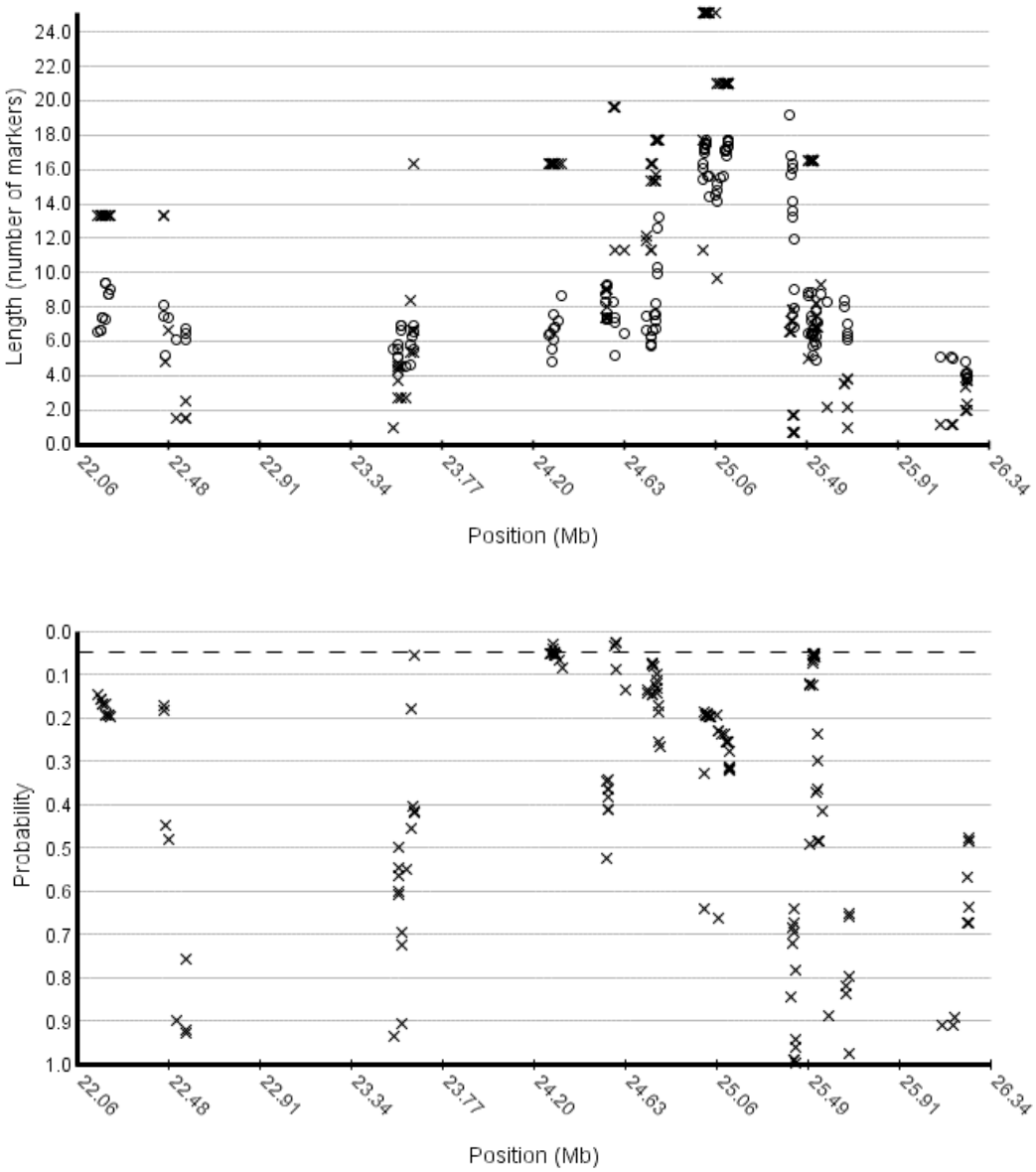
Region D was the second of the high priority linkage regions and was the most statistically significant (in terms of LOD scores). Three families (F22, F50 and F48) showed linkage to the region. One of these families was Scottish, one was Welsh and the other was an American family of Ashkenazi Jewish descent.

4.3.1 Phase I: Allele sharing at known genes

As with *region B*, the initial analysis of *region D* only included markers about some of the genes in the region. Three *disease-linked* and 37 *control* haplotypes were tested for excess allele sharing at these 157 markers. These results have been published in le Hellard et al. (2006).

Allele sharing analysis identified two sub-haplotypes that showed a shared haplotype between the three *disease-linked* haplotypes and also contained some markers that were significant on an individual marker basis. The first haplotype, consisting of 10 markers over two LD blocks (over 55kb) showed higher sharing between disease and control chromosomes (see Figure 4.4-A). Permutation analysis showed that two of these markers had a P value < 0.05 while the others were not deemed significant. The average P value for the shared region was P=0.054 (Figure 4.4-B). After the secondary permutation analysis this region was clearly not significant (corrected P value P=0.52). The second haplotype consists of three markers over one LD block (over 9kb), all three markers are significant and the average P value was 0.031. However, the corrected P value for this region is 0.416.

Figure 4.6: Phase I: Allele sharing in *priority region D.*



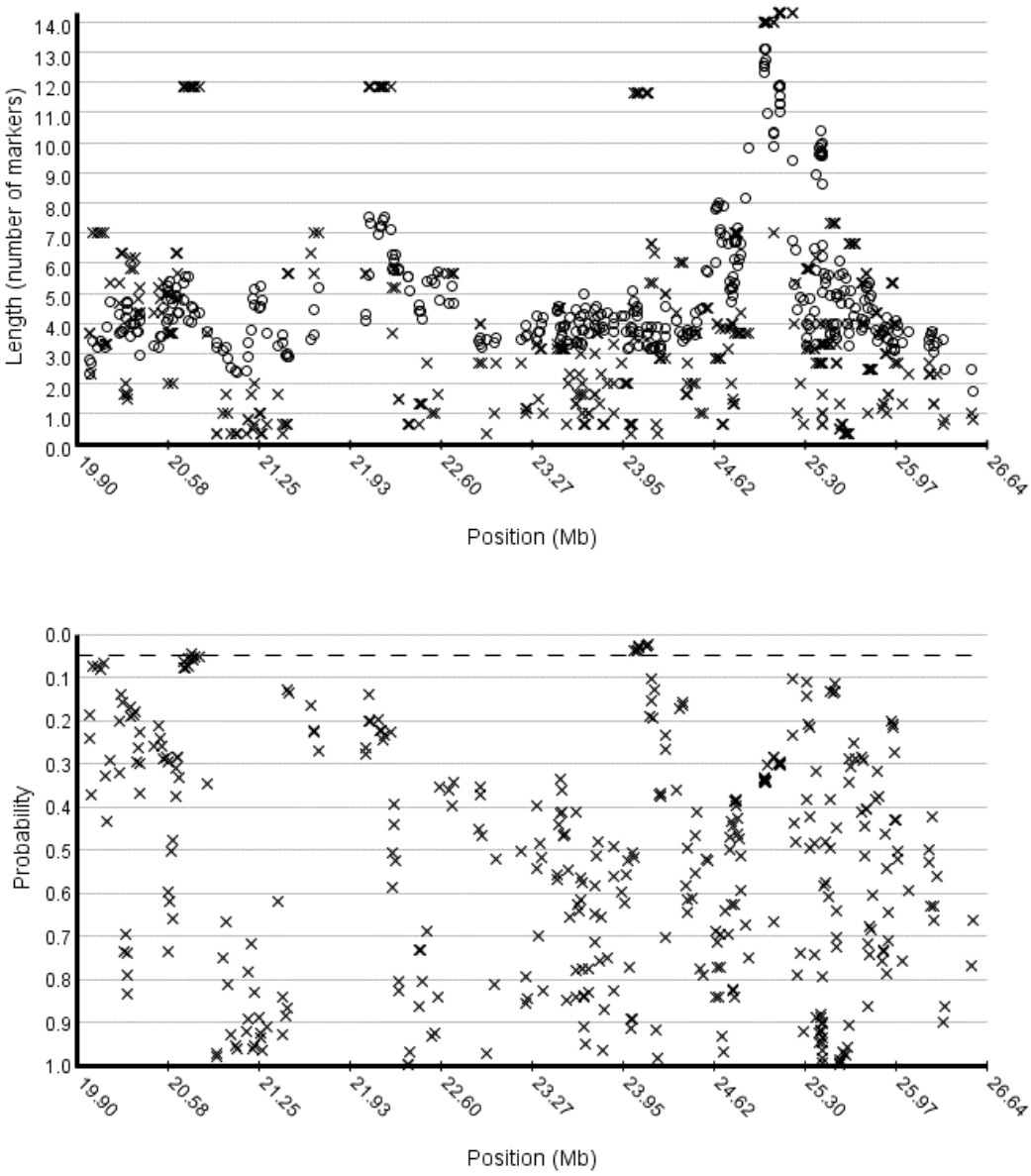
Shows allele sharing for markers genotyped around genic regions of region B. (A) Allele sharing scored and (B) significance values generated through permutation analysis.

4.3.2 Phase II: Allele sharing across the region using haplotype-tagging markers

In phase II of the analysis of *priority region D*, 384 markers were genotyped across the region. The disease-linked and control haplotypes were tested accordingly. Figure 4.5-A shows the allele sharing scores in the *region D* where there appears to be four peaks in the allele sharing across the region for both *disease-linked* and *control* haplotypes. Although the peaks are higher in the disease-linked haplotypes, there are only two clusters of markers that are significant on an individual marker basis.

There also appear to be a number of regions where the sharing is higher between the disease-linked and control haplotypes. However, the differences are not as high as in the *priority region B* and seem to match to regions where the sharing between control haplotypes peaks. Upon carrying out the permutation analysis, two shared regions were found to carry at least one significant marker (Figure 4.5(B); Table 4.2). However, none of these regions remains significant after correcting for multiple testing.

Figure 4.7: Phase II: Allele sharing in *priority region D.*



Allele sharing for markers tagging haplotypes across all of region B. (A) Allele sharing scores and (B) significance values determines through permutation analysis.

Table 4.2. *Priority Region D* shared regions.

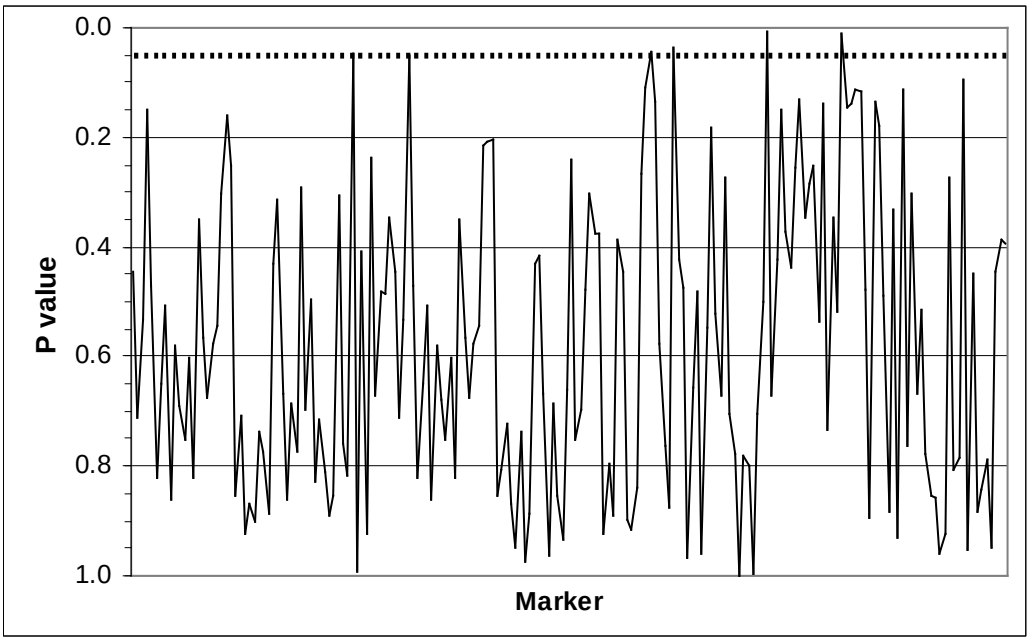
Shared region	No. shared markers	No. LD blocks shared	Length of shared region (kb)	*P* value	Corrected *P* value	Genomic location
12	10	9	117	0.062	0.520	20,556,005 – 20,672,923
13	11	10	101	0.031	0.416	23,898,753 – 23,999,911

The table lists the properties of the regions of excess sharing in priority region D of the F22 linkage region, the number of shared markers within them, the number of LD blocks they cover, their length, the P value after the test for significance, the P value corrected for multiple testing and their genomic location (chromosome 4 NCBI Build 36 March 2006 assembly coordinates).

4.4 TDT analysis of priority region B

Region B was the location of a region of allele sharing that was found to be significant based on the allele sharing method developed in this thesis. The WHAP program was used to provide a more traditional test of association in *region B.* Ten parent offspring trios were selected from the three families showing linkage to the region (F22, F59 and F50). The first test was an individual test of each marker in the region and was run with the default WHAP settings and significance tested through 1,000 permutations. The distribution of P values across the region is shown in figure 4.7 below. The two markers with highest level of significance (P=0.006, P=0.010) are both found within *shared region 1* which was found to be significant through the allele sharing analysis. Four other markers showed marginal significance in regions that also corresponded to peaks in allele sharing. Two-marker haplotype analysis was carried out and showed similar results. The 17 marker haplotype was also tested and was found to be very significant ($P=1.85x10^{-33}$).

Figure 4.8: Single marker association in *priority region B.*

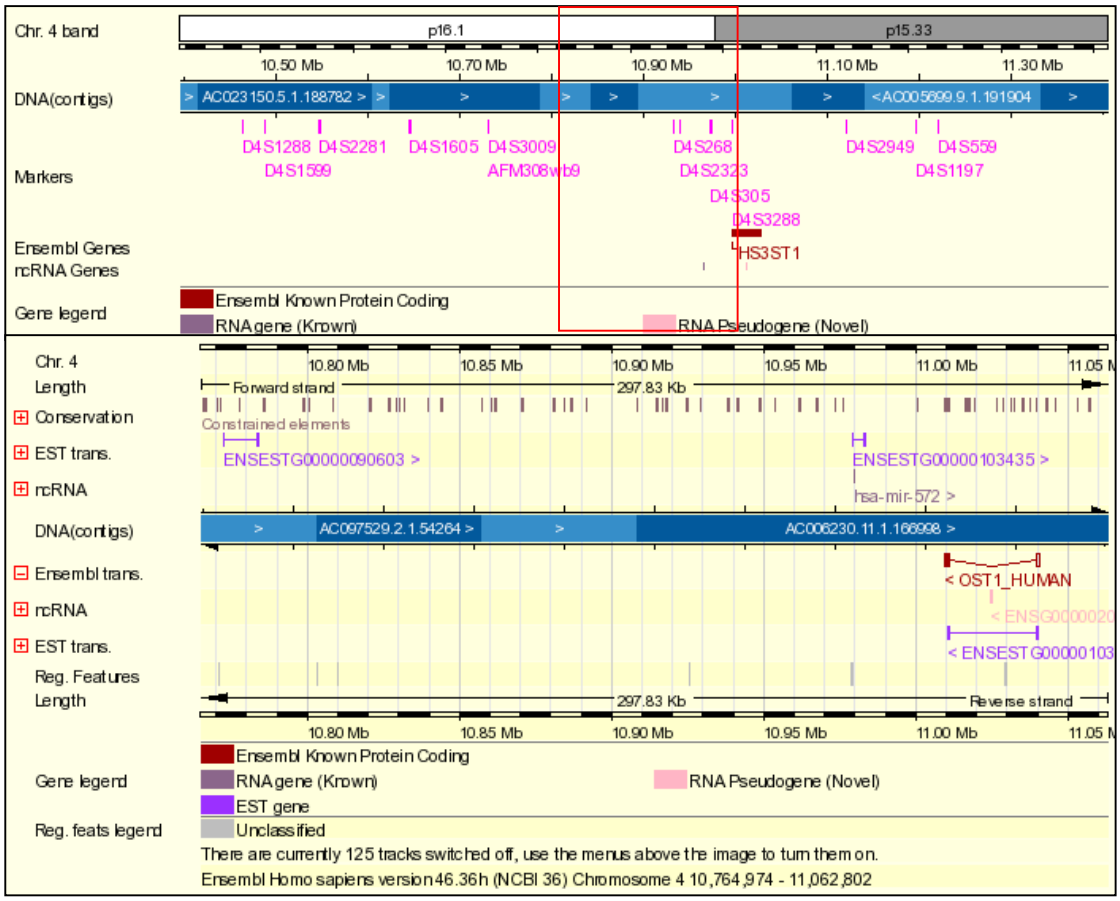


Significance of each marker based on single maker association analysis

4.5 An investigation of a significant shared region

Bioinformatics tools were used to assess the biological significance of the significant shared regions. This focussed on *shared region 1* found in *priority region B* (section 4.2.2), which was the most significant region found through the allele sharing analysis based on the complete genotyping coverage. Figure 4.8 shows the Ensembl ContigView of the region.

Figure 4.9: *Shared region 1* on Ensembl ContigView



This shows the annotation of shared region 1 in priority region B, showing the OST1 gene (HS3ST1).

One known gene is found in this shared region, *heparan sulfate (glucosamine) 3-O-sulfotransferase 1* (HS3ST1; MIM 603244). The HS3ST gene is highly expressed in the brain and kidney and weakly expressed in the heart, lung and placenta. It possesses both heparan sulfate glucosaminyl 3-O-sulfotransferase activity and anticoagulant heparan sulfate conversion activity, is a rate limiting enzyme for

synthesis of anticoagulant heparin and is an intraluminal Golgi resident protein. However, HS3ST1-deficient mice did not show the expected anticoagulant phenotype. Instead, they developed unexpected non-thrombotic, perinatal phenotypes, including eye degeneration and postnatal lethality. This suggests that the HS3ST1 enzyme might have additional or alternative biological roles. There is a also an miRNA sequence, hs-mir-572, a non-human RefSeq gene and one main class gene predicted by Aceview within the shared region, as well as number of spliced ESTs and regions of multi-species conservation.

4.6 Discussion

The evidence for a susceptibility or causative locus or loci in the chromosome 4p15-16 region is strong. However, due to the complex genetic nature of the disorder, it has proven difficult to pin point any such loci. In this chapter I have reported on a study of allele sharing in the region, investigating the possibility that a common founder mutation is shared between the families.

Figure 4.3 shows the genomic location at which four particular families display linkage with BPAD and BP-related illness. Two priority regions were identified where the linkage signals of three of the four families overlapped. The fact that two sub-regions were identified may indicate the presence of two susceptibility loci, one in *priority region B* and one in *priority region D*. In each of the two sub-regions, it is possible that the three families that show linkage do so because they share the same mutation and that mutation has been inherited from a common ancestor. However, it may be the case that some of the families have inherited the same mutation, but from a different source, meaning that they would not share the same genetic background around the mutation and it is also possible that any of the families could carry more than one mutation in the region. *Priority region B* is made up from three families (two Scottish and one Welsh) thought to be of Celtic origin, so it is particularly plausible that an ancestral mutation could be shared by these three families. The second priority region (*D*) is made up of a Scottish, Welsh and US family of Ashkenazi Jewish origin. A mutation of common ancestral origin would most likely have to be very old to have occurred in a lineage shared by these three families.

The initial investigation into allele sharing in the region relied on genotype data only from markers in the areas around known genes. This clearly limited the analysis by leaving substantial areas in the regions untested; it may also have had a negative impact on the scoring system due to some wide gaps between some markers. This analysis did uncover one significant region, with a P value of 0.009, in *priority region B*. It was clear that the results would be uncertain while there was incomplete genotype coverage of the region. The allele sharing peaks at this shared region in the *disease-linked* haplotyped, but it also peaks in the level of sharing between the *control* haplotypes. The increased sharing in the disease-linked group may just be an artefact

due to there being only three families in that group, compared with 38 in the *control* group, giving a broader picture of haplotype diversity in the population.

In the second phase of the study, markers were selected to encompass the LD structure across the two priority regions. This was intended to provide an almost complete coverage of the region, where a marker was genotyped that would cover any haplotype block that occurred with 10% or greater frequency in the CEPH trios. This should have made the scoring system much more consistent as the spread of markers was even across the LD variation map of the region. The most interesting results lay in the large non-genic region that had not previously been genotyped. Here, a large extended region of 71 markers was found in which all the makers were significant on an individual marker basis. There was found to be consistently low allele sharing between control haplotypes and high allele sharing between disease-linked haplotypes across this region. The region that was previously found to be significant in region B following the phase one analysis, still corresponded to peak in the allele sharing scores, although it was no longer found to be significant after multiple testing was taken into account. In region D, there was one individual marker and another cluster of 11 markers that were significant.

After the correction for multiple testing, there were 11 regions that are found to be significant. All of these were in *region B*, and in common with the phase I study, there were no significant regions found in *region D*. These 11 regions ranged in size from 1 marker to 18 markers; accounting for, between 1 LD block and 17 LD blocks; spreading across up to 197kb; and with P values ranging from 0.01 to 0.005 on an individual marker basis and between 0.05 and 0.008 after correcting for multiple testing. One region in particular stood out, *shared region 1* covers a much larger region than the other significant regions (17 LD blocks compared with 1-6 LD blocks in the other regions) and is much more significant than the others (P = 0.008 compared with P = 0.026-0.05).

It is also useful to compare the results of the BP-linked families with those of the simulation study. We can look for the simulations that present similar results to those of the chromosome 4p study. Here we find that if three families are generated with a similar density of markers genotyped and with an average of around 400 generations between the families, then we see a similar pattern of results in terms of the size of the shared regions identified and their associated P values. The analysis of simulated data

of this form tended to identify one relatively large region that was very significant which would contain the mutation and a number of smaller regions that were of more marginal significance that were false positives. It was also important to consider the anomalies identified in the false positive rates from the simulation study. It was clear from the results of the simulation study that the P values being reported were not always accurately reflecting the genuine significance value. However, it was possible to estimate what sort of adjustment to the P values being reported in *shared region I* were necessary. Looking at the range of simulation results that the results reported for *shared region I* fall into, then we might expect that P values are being over-estimated. The closest category studied in the simulation corresponding to the results described above suggest that correcting P values by a factor of about 5 would be more accurate representation of probability values. This would still suggest that the 197kb region was a true positive results within which we would expect to find a mutation. Although clearly, this issue will cast doubt on this result, there was some evidence that this result is genuine.

We can also use the comparison with the simulation study to estimate the age of the mutation. The results, in terms of the significance values, number of false positives etc, match most closely with those simulations where there were ~400 generations between the families being tested and a common ancestor. A very simple approximation could put this at something like ~8000 years separation. Of course this would be further complicated by the fact that two of the families (Scottish families F22 and F59) are likely to share a much more recent common ancestor than they do with the third (Welsh family F50). It does give a broad outline of the region we are looking at. The fact that it is *region B* within which we find a region of significant sharing and not region D backs up our original assumption that it would be much more likely that the three families of Celtic origin would share a common founder mutation than two Celtic families with a family of Ashkenazi Jewish descent. This hypothesis is supported by a closer study of the shared haplotype between the three families in *priority region B.* All thee families share the haplotype that forms *shared region 1*, the two Scottish families (F22 and F59) share a haplotype that extends far beyond *shared region 1*.

There is one known gene in s*hared region 1*, where the significant region overlaps with 5kb of HS3ST1 (MIM 603244). Although HS3ST1 is not an obvious candidate

for BPAD, it is highly expressed in the brain and recent studies on knock-out mice have shown that it may have a much more diverse function than just as a rate limiting enzyme for synthesis of anticoagulant heparin. Some further evidence to back up HS3ST1 as a candidate disease gene has come from the PROSPECTR tool (Adie et al. 2005) which found HS3ST1 to be the most likely candidate to be a disease gene out of all the genes in the two priority regions. PROSPECTR works on the assumption that certain sequence based features such as larger gene lengths and broader conservation through evolution can indicate that a gene is more likely to be implicated in disease. It found that HS3ST1 was ~2.1 times more likely to be implicated with disease than a random gene. The majority of *shared region 1* consists of 193kb upstream of HS3ST1. In this region are a number of predicted genetic features and are likely to be regulatory elements relating to HS3ST1. Further evidence for this region has been reported recently. Christoforou et al. (2007; section 4.1) described an association study where they found a haplotype in *shared region 1* that was significantly associated with Schizophrenia (uncorrected P value = 0.00046; OR = 2.2; 95% CI: 1.3-3.6) in a Scottish cohort.

Further evidence for the *shared region 1* comes form the association analysis of region B where the two most significant results of the single marker analysis are found. Unfortunately, this is the limit of the analysis that was carried out using existing published methods. As discussed in the introduction, it is very important that the new allele sharing method described in this thesis be tested against existing published methods, unfortunately this proved difficult. Those haplotype based methods similar to the one described in this thesis were not found to be readily available and therefore I was not able to test the chromosome 4p data using them. Equally, other haplotype sharing methods were found difficult to find available programs that allowed them to be run with the data used in this thesis. TDT is perhaps a more established method and there are many versions implemented in many different software packages that are easily available. They were not always found to be the most intuitive nor with the most helpful documentation. There was also a problem when it came to running this analysis on a personal computer that failed to carryout much more than the basic single marker analysis.

This situation leaves some important work that needs to take place before the above method can be taken forward. Especially in light of the problems encountered with

the false positive rate. It may also reflect the situation in the field whereby the motivation of those developing new methods is not to develop a polished software program, but rather to aid their own research. It is a similar situation that led to the development of the allele sharing method described in this thesis as it was felt that existing methods were difficult to apply to the scenario that I encountered.

In summary, this study has shown how genotype data from families presenting a common linked region can be used to greater effect. Here an initial linkage result of 20Mb was reported in the family F22 (Blackwood et al. 1996), this was reduced to 8.1Mb (4.3Mb and 3.8Mb) through studying the overlap between F22 and three other families (F59, F50 and F48) that also showed linkage to this region (Le Hellard et al. 2006). I have now shown how the allele sharing method presented in this thesis has been used to reduce the region even further, to 197kb. *Shared region 1* should now be prioritised for further study.

Chapter 5

Discussion and Conclusions

One of the key challenges in the field of medical genetics is to identify ‘disease genes’ or loci that cause or confer susceptibility to disease. The traditional approach of genetic linkage and positional cloning has not achieved the same success for complex disorders as it has for simple single gene diseases. While new approaches such as large scale and genome wide association studies and copy number variation analysis may provide new avenues for mapping disease loci, genetic linkage can still provide very good evidence that a relatively large genomic region contains a susceptibility locus. While more traditional family based approached exist to test for association, haplotype sharing statistics have been shown, in at least some circumstances, to have greater power in identifying genetic associations with disease (Tzeng et al. 2003; Allen and Satten 2007). In this thesis I have described and applied a method for investigating allele (or haplotype) sharing in large families, with common regions of linkage, to identify a common founder mutation.

5.1 A method for studying allele sharing

In the case of the extended chromosome 4p linked families studied in this thesis, families were identified for study having shown to segregate with BPAD through linkage analysis. Study of genotype data from these families then clearly showed one haplotype that was carried by almost all affected individuals. It is to be expected that a large family that shows significant linkage for a region, that those individuals displaying the trait being studied should share a haplotype across that region. It was the question of how to analyse such data that lead to this thesis.

Family based methods for investigating association exist with many modification of the original transmission disequilibrium test (TDT) of Spielman et al. (1993), but there is greater value to be found by incorporating the information of the haplotype into such analysis. Extensions to traditional family based association studies to allow haplotype information to be used have been developed, however a newer type of analysis based on shared haplotypes has become popular. A number of methods for measuring allele (or haplotype) sharing were discussed in the introduction to this thesis. While it proved not to be possible to apply any of the existing methods directly to the family data such as that generated from the BP-related families linked with chromosome 4p15-16, the scoring system developed by Van der Meulen & te Meerman (1997) and Bourgain et al. (2000, 2001, 2002) provided a good basis from which to develop a new method. The initial concept of carrying out a pairwise comparison of groups of case and control haplotypes and system for scoring the similarity of a pair of haplotypes were sound. The aim of this thesis was to develop a method, based on the principles developed for other haplotype sharing methods, that was suitable for the study of multiple, large family and large scale genotyping data such as that produced from the chromosome 4p linked BPAD families.

The basic scoring system involved counting the number of markers shared consecutively between haplotypes (Bourgain et al. 2000). An alternative to using the number of markers in a shared region would have been to score sharing based on the actual size (in base pairs) of the region thus accounting for the distribution of the markers. An initial investigation suggested that the simple counting scoring system generated sounder results and was more robust. It was thought that the scores obtained by the incorporating the actual distance between markers were too strongly influenced by the uneven distribution of markers across the regions under

investigation. In the case of the BPAD-linked families, there were very large regions between markers (as a consequence of the LD structure in the region) and this gave some markers too much weight. There was also a choice of methods to manage ambiguous and missing data. The method used for dealing with missing data is logical, given that it is impossible to say whether that marker is shared or not shared between haplotypes, it can be ignored, and it will be possible for a shared region to extend around it while that marker will not contribute to the total number of markers in that region that are used to form the score. Ambiguous data is dealt with in a similar way, with a sharing region allowed to extend around an ambiguous marker so long as one of the potential allele combinations does mean a match between the two haplotypes. Unlike the missing data, ambiguous data *will* contribute to the score of the region, this score it contributes is in proportion with the likelihood that the maker is expected to be shared. A brief investigation indicated that this has limited effect the results although it might be expected to have greater effect if the level of missing or ambiguous data in much greater that in the samples reported in this thesis.

Bourgain et al. (2000) also developed a test to determine statistical significance based on their allele sharing measure. Their test compared the maximum difference in sharing between transmitted and non-transmitted haplotypes to that of a series of simulated haplotypes with no difference. While a useful statistic for identifying a region as significant, it has the limitation of being unable to investigate the variation in sharing across a region and the possibility of identifying a sub-region. It is also uncertain how appropriate their group of haplotypes that were constructed to be used in this test are. Tzeng et al. (2003a) showed just how difficult a job it is to model the statistical complexity of this type of method. I developed a method of permutation analysis to test the statistical significance on an individual marker basis and then nested permutation analysis to account for the multiple testing. By permuting randomly reassigned disease-linked and control haplotypes, it is assured that differences in haplotype structure will not confound the result. Nested permutations, to generate a modified P value, which accounts for multiple testing, for any significant region will similarly benefit from using a same genetic background. The computational requirements of nested permutations are high and a method of approximating nested permutations by reusing the permutations generated in the initial permutation analysis was used. This method was proposed by Ge et al. (2003) to generate corrected significance values for microarray analysis and was later used

by Becker & Knapp (2004) to generate corrected significance values for association fine mapping. Both showed that the significant computational and time gains could be made with little loss of power. While the initial permutation analysis was concerned with comparing individual markers, the nested permutation analysis was interested in generating a corrected significance value for a *region* that had been identified initially as significant. For this process, it was important to take the size of the region into account. Rather than physical size, the number of LD blocks that a region encompassed was used. This reflected the lack of independent inheritance between markers in strong LD with each other. So, the nested permutation analysis is asking the question of whether any sub-region anywhere in the wider region under study is as significant over the same size of region (in LD blocks) as the one which the corrected significance value is being calculated. The LD structure for the BPAD linked regions under study in this thesis was defined using data from HapMap Release 7 (May 2004; http://www.hapmap.org). This data is based on the 30 CEPH trios who were Utah residents with ancestry from northern and western Europe (CEU) and so should form a reasonable sample from which to make conclusions regarding the LD structure of northern European and Ashkenazi Jewish families that have been studied in this thesis.

Since the work of Bourgain et al was published (2000, 2001, 2002), the body of literature in the area of haplotype sharing methods has grown substantially with methods being published based on a variety of approaches. While many of these methods produced tests for significance, often relying on permutation analysis, there has been some attempt to produce a common statistical framework for these methods. Unfortunately, the work reported in this thesis to apply statistical test based on permutation analysis was undertaken some time ago and was also carried out in an 'ad-hoc manner' like many of the other methods and it fails to take advantage of the work done to develop such a statistical framework. It is likely to be due to this ad-hoc development that the false positive rate is found to be problematic.

It is also important to consider such an analysis as part of a range of tools available for studying family data. While some have concluded that haplotype sharing methods are more powerful than more traditional association type methods, Tzeng et al (2003a) and Klei & Roeder (2007) both show that there is a lack of correlation in power between the two main approaches to studying allele sharing. Even if this is not the case, it appears to be sensible to pursue a multi-method strategy in analysing such

data. Only very superficial analysis was carried out using more established TDT method, but even that appeared to support the results of the allele sharing analysis.

Further development of this method, however, could also allow it to be used in wider circumstances. Of course, some of the general issues that arise in the use of haplotype sharing and association study methods will still be of concern. Population stratification and cryptic relatedness are generally not a problem in family based studies, however, between-family studies like that proposed here could suffer from unknown relationships between the families. In the specific study of the Chr. 4p linked families they are well defined and very geographically disparate, so this is unlikely to be an issue, but it should be kept in consideration. Power may also be an issue, given that there are not necessarily large numbers of families that would be available in the scenario outlined above, indeed the Chr. 4p linked region study provides just three families in two regions of study and two of the families are smaller families displaying only supportive LOD scores. This provides good reason to pursue multiple strategies in investigating the linkage region with the allele sharing method just one of these.

5.2 Testing and proving the method

Although I felt I had developed the method on a sound theoretical basis, before applying it to experimental data, I felt that it was important to provide evidence for its efficacy in identifying founder mutation carrying haplotypes between families. This proved to be challenging, as there are few cases where there are a number of families known to carry the same disease susceptibility or disease causing mutation and that have been genotyped to a significant degree and it would have been a very expensive undertaking to genotype families with a known mutation just for the purpose of this study. Data was eventually sourced that proved adequate to test most aspects of the method.

5.2.1 Cystic fibrosis study

This data was consisted genotype data from individuals of 60 Breton families known to carry any one of three cystic fibrosis mutations. The Breton population is thought to be fairly distinct and relatively isolated from the rest of France and it considered very possible that a number of the families carrying the same mutation would do so courtesy of a common ancestor and that this would be detectable through the study of allele sharing around these mutations (De Braekeleer et al. 1996). Individuals from these families had been genotyped at 10 markers across an 8Mb region encompassing the CFTR locus that carried the cystic fibrosis mutations. While it would have been preferable to have a much more dense genotyping of the region, it was felt that given the large number of families, if enough of them shared a common ancestor that was recent enough, the we should be able to detect a shared region that extends across multiple of these markers. The analysis itself showed that almost all the markers were significant on an individual marker basis. However, there was a clear peak, in each of the three cases, at the markers around the CFTR locus, where the sharing was much more significant than the rest of the region. It was not possible to use the nested permutation analysis to generate a corrected significance value for any shared region as there were no haplotypes that were consistently shared by all *disease-linked* haplotypes. This was most likely due to there being a large number of families which may have brought in some families from a different genetic background; the sparse genotyping meant that it was unlikely that all the families would share a consistent

haplotype across such a large region; finally, the markers studied were microsatellites, which are known to be much more mutable than SNPs. However, due to the small number of markers being tested, it was felt that a Bonferroni correction would be a feasible approach to take. Following the Bonferroni correction, in the case of each mutation, the region formed by the markers flanking the mutations was found to remain significant, while the rest of the markers fell either into insignificance or marginal significance.

The analysis of these families is slightly unnatural due to the families being split into groups based on which mutation if carried by the affected individuals. This analysis, however, is still appropriate in that it was designed to prove the case whereby the allele sharing method would identify the common mutation carrying region where it is known to exist. However, in the more realistic scenario where these families would have been mixed up together, the method does still find the those markers flanking the mutations to be the most significant, albeit at a much reduced level than in the three subgroups (data not shown). It is clear that this might not always be the case. To avoid positive results being lost by the inclusion of families that do not share a common haplotype around a founder mutation, it should be possible to study allele sharing between different combinations of families in order to identify whether any particular combination of families produce a particularly significant result. This would, of course, increase the number of test being carried out and this would have to be dealt with on assessing true significance. This would be a very useful area of future research that might allow the method to be used more fully.

Although the data studied here was not ideal, the study of these three cystic fibrosis datasets does provide a clear example of the method being successful in identifying a region surrounding a known disease causing mutation. This backs up the assumption that these mutation lie on some shared ancestral haplotype that was inherited by most of the families used in this study.

5.2.2 Simulated data study

To test the method further, a simulation was developed to generate distantly related families that carry a shared mutation. An initial mutation and its expansion (or otherwise) into a population was simulated. After a number of generations, families could be identified on which to test the allele sharing method. Although the

simulation method assumed that there was 100% penetrance and no allelic heterogeneity, it was felt that these were fair assumptions as far as the purpose of this study was concerned. The initial premise for the use of the allele sharing method was that families had been identified (probably though a linkage study) showing a genetic region that segregates with a disease or trait. This was not something that was simulated, rather the assumption was made that for it to have been possible to identify the simulated families prior to testing the allele sharing method, it is unlikely that these cases would display such linkage should the not show high penetrance and low levels of allelic heterogeneity. It was therefore deemed acceptable, within this basic model, that the best case scenario be taken as even in cases of high, but less than 100%, penetrance and low levels, but non zero, allelic heterogeneity, it should still be possible to identify a common haplotype found in most affected individuals in our simulated families and therefore identify a *disease linked* haplotype.

The simulation study did allow for a number of more relevant scenarios to be modelled. The method was tested using various numbers of families, genotyped at various different densities, while varying the number of generations between the families and a common ancestor. These test showed that for as few as three families, where genotyping was carried out to a sufficient density (~one marker every 11kb), all but the most ancient ancestral haplotypes would be expected to be detected. It was unfortunate that the computational limitations of the simulation meant that the simulation could not be extended for more than 410 generations (8250 years) between the simulated families and their common ancestor, however, the trends were clear. The data presented in Table 3.2 can be used as a rough guide to show the density of genotyping required in a study. While the simulation study showed that the allele sharing method proved to be highly successful at identifying the simulated mutation carrying region under a variety of conditions, the difference between the expected and observed false positive rates casts doubt on the accuracy of the significance values. This would appear to be due to a fundamental error in the way the significance values are generated and will need a thorough re-evaluation of these methods.

Another aspect that arose from the study of the simulated families was how the results could be used to compare against the results from novel studies. Data was generated in a wide range of condition, by comparing the results of a novel study with those of the simulated data, it is possible to draw conclusions on aspects of the result, such as the nature of true and false positives and of the age of any ancestral haplotype block

discovered. However, due to the simplistic nature of the simulation, this may not stand up to scrutiny.

These two avenues were pursued to develop confidence in the ability of the allele sharing method to accurately detect the ancestral haplotype expected to flank a mutation inherited from some common ancestor. Although the CF study data is not quite similar to the data that this method was created to study and the simulation study is very simplistic, I feel that these studies have proven that the method is successful in identifying mutations under a range of conditions. However, it has also thrown up some worrying features, namely the unexpected false positive rate. While this is something that creates doubt as to the veracity of the method and the significance values it generates, and this is something that should be resolved, it should be negate the value that the use of the method can provide. It does mean that the results reported in chapter 4 of this thesis should be treated with some caution.

5.3 Implementation of the method

All the methods described here were developed as a suite of programs. These programs included the allele sharing methods themselves, a program to present the output of the allele sharing scoring and permutation analysis visually, a program to generate the simulated data and test it and many other programs relating to the necessary data manipulation and formatting. The programs were developed in Java primarily as it was the language I was most familiar with and was perfectly suitable for these tasks, but Java also had the advantage of meaning that the programs are portable across platforms. This work, in particular the computationally intensive simulations, was conducted on a number of different machines running Windows, Linux and Solaris platforms, so it proved very useful to be able to transfer programs without needing to significantly recode them.

While it would have been preferable to have designed the programs completely prior to coding them, due to the nature of the way the work developed (i.e. development continued even while the initial analysis had begun), the programs were developed in a piecemeal manner. As a consequence, there are many features that should be easily variable to the user that were hard coded, meaning that the code must be altered to change some aspects of how the program runs. While this code is available publicly, it would be preferable to create a more polished executable that does not require the source code to be visible at all. There are other aspects of the programs that could also be improved, such as merging the programs that calculate the allele sharing scores and initial permutation analysis with the program that carries out the nested permutation analysis that corrects for multiple testing. It would also be possible to automate certain aspects that are required to be carried out by hand at present, such as the identification of shared blocks and the incorporation of the haplotype structure of a region. Both these features were incorporated into the simulation study, which was almost fully automated, so much of this code could be reused to create a more complete program for the study of allele sharing.

The most relevant classes and methods have been included in this thesis, either in sections 2.1.5 and 2.3.2 or in Appendix A.

5.4 Testing the allele sharing method and applying it to BPAD linked families

The four families that present genetic linkage with BPAD and BP-related illness at the chromosome 4p15-16 locus (Blackwood et al. 1996, Detera-Wadleigh et al. 1999, Asherson et al. 1998) provided the initial motivation for developing the allele sharing method described in this thesis. These four families that showed strong evidence for the same location being implicated in major mental illness, yet the pace of progress in identifying the cause of those linkage signals was slow. Some candidate gene studies in the region had shown little more than slight evidence for an association with BPAD and there were no other obvious candidates amongst a number of genes in the region that would be plausible to related to BPAD. Allele sharing analysis developed in this thesis was intended to provide a new use for the families that generated the initial linkage results and to use them to narrow down the region under investigation even further.

The original linkage results in F22 highlighted a region of ~20Mb (Blackwood et al. 1996), this was reduced to two priority regions encompassing ~8Mb as additional linkage results were taken into account (Le Hellard et al. 2006). The allele sharing analysis aimed to reduce that further more. The families were genotyped and studied in two phases, with the second phase providing a high level genotypic description of two priority regions. *Disease-linked* and *control* haplotypes from the four families were compared in the two regions and a number of significant results were found, with one significant shared region in particular standing out, *shared region 1*. This region covered 197kb, 17 LD blocks (as defined by the CEPH trios in haploview) and was significant after correcting for multiple testing with a P value = 0.009. While this appears to show strong evidence for a susceptibility or causative mutation to be found in this region, it is worth considering how accurate the reported P value is. The simulation study showed that there was a problem with the false positive rate, casting doubt on the veracity of the significance values being generated. Further analysis of the simulation study data suggested that accuracy of p values varied widely. It was possible to show that those simulation that were most similar to the chromosome 4p linked family study showed an overestimation of P values of about an order of 5. It is unfortunate that such doubt has been cast on what appears to be a very positive result.

It has to be hoped that, through the redesign of the significance testing or through further investigation of the simulated datasets, that this issue can be resolved and that the reported P values can be believed with greater confidence.

It is hoped that the issue of uncertainty relating to the significance values does not detract too much from the value of the results reported here. These results still provide evidence for a substantially reduced sub region that should be prioritised for further research. These results should also demonstrate how such a method should be able to similarly allow shared haplotypes to be used to generate priority sub regions of large candidate regions where the appropriate family data exists. However, it is important that allele/haplotype sharing methods are used as part of a mixed set of tests as they all have different strengths and weaknesses and it is not always easy to tell which method will be most successful for a particular study.

5.5 Final conclusions

Genetic linkage can provide very good evidence for a susceptibility locus, but this is usually over a relatively large genomic region. While the results of linkage studies have proven useful where simple Mendelian disorders are concerned, for complex disease, it has proven difficult to make progress from this point. Therefore one of the main challenges facing those tackling the genetic analysis of complex disease is of how best to capitalise on linkage results without having to scan the entire region for association. In cases of large families showing replicated linkage thought to be due to an ancestral mutation in common between the families, the method presented in this thesis addresses this problem through the investigation of allele sharing throughout the linked region(s). Although it might appear quite limiting for this method to be restricted to quite a specific scenario, this situation is increasing common and there appears to be a need for such a method in this area. Complicating issues such as population stratification and cryptic relatedness, add to the reasons for continuing to pursue a family based approach in narrowing down such linkage regions in the pursuit of disease causing loci. Allele, or haplotype, sharing methods such as the one described in this thesis attempt to use the value stored in the haplotypes of families under study to identify a region linked with a disease mutation. There is also no reason why this method cannot also be extended for use in other areas, although perhaps existing methods are more appropriate. While there are a number of published methods that have looked at how alleles of haplotypes can be compared, none of them have been developed into a practical method that could carry out the specific role required for this, and similar, studies.

In chapter 3 I have shown extensive evidence that the method developed here is effective in identifying an ancestral region for most founder mutations. I have also identified a set of criteria that shows how its efficacy varies under different conditions. In many cases, it will prove to be the most cost-effective approach to genotype and test for allele sharing where a founder mutation is expected, as this could reduce substantially the region for fine-scale (sequence level) investigation. In reality, allele sharing will probably be used alongside other tools. This work has also identified some discrepancies in the false positive rate. While the method is shown to be able to accurately identify the mutation carrying region through simulation, the

accuracy of P values is in doubt. Investigation into the variation in the false positive rate allowed some estimation of the error involved, but this is a problem that needs to be solved through the redesign of the statistical test itself.

Chapter 4 describes the application of the methods described here to a genomic region that has shown evidence in a number of families of linkage to BPAD. This study detected a number of significant regions, but on comparison with the results of simulated data, on region of 197kb appears to stand out. It is hoped that this region will undergo further study to uncover the likely mutation. It is expected that this scenario where a number of families are linked to the same region will becoming more common in studies of complex disease and the process described in chapter 4 describes how these allele sharing methods can be used to combine these linkage studies in order to make progress in the search for a mutation or causative locus.

While future advances in technology (namely cheap rapid large scale sequencing) may lead to linkage based methods being completely superseded. However at present this method should be a valuable addition to the tools available in the study of the genetics of complex disorders such as bipolar affective disorder.

5.6 Summary

The following is a summary of the main achievements of the work presented in this thesis:

- Identified a gap in the positional cloning process for complex disorders;
- Developed a method for studying allele sharing that could help bridge this gap, building upon the existing published body of work and extending it to become allow application to a specific, but not uncommon, scenario;
- Presented evidence to show that this method was effective in taking genotype data from families carrying three CFTR mutations responsible for cystic fibrosis and used this data to identify the region that contains these mutations;
- Developed a basic simulation that models how a founder mutation developed in a population through a variable number of generations;
- Presented evidence to show that the allele sharing method was effective in identifying the mutations in this simulated data under most normal conditions of the simulation (i.e. a study of at least three families, where there is less than 400 generations separating the families from a common ancestor);
- Presented the results from a wide range of simulated data that shows how the allele sharing method would be expected to perform under a variety of conditions and characterised the variation in the false positive rate;
- Showed how the allele sharing method has been applied to data from a number of families showing linkage to chromosome 4p15-16 for bipolar affective disorder to identify a sub-region within the existing linkage region;
- Described how these results can provide the basis for more focussed research into the cause of this linkage signal;
- Developed an initial insight into the age of the BPAD mutation carried by the 4p-linked families;
- Described how the study of these chromosome 4p15-16 linked families can be used as a basis for getting greater use out of large families that show replicated linkage with complex genetic disease.

5.7 Further work and recommendations

In order to progress this work further, there is one major issue that needs to be resolved, that of the false positives identified in the analysis of the simulated data. There is clearly some feature of the test being carried out that is leading to an unexpected false positive rate that needs to be identified and corrected. Until this is carried out any results based on this method will have some doubt case upon them. If the issue of false positives cannot be resolved, alternative methods based on more recent work should be used. Provided the issue of false positives is resolved one way or another, it would be useful to look at the following areas:

Allele sharing method and program

- A much more thorough comparison between the results of the method described in this thesis and other existing methods is required;
- It may be worth testing the effect of different scoring systems;
- It would be very useful to have additional datasets upon which to test the method;
- I would like to implement a more 'user friendly' version of the program so that it could be more widely used;
- Investigate how the method could be extended for use in testing for allele sharing between unrelated cases;
- Provide a mechanism and statistical basis for the method to be used in sub-combinations of families.

Simulation study

- Increase the computational power available to allow the datasets that can be simulated to be expanded;
- Improve the simulation to allow more variability in the manner of population expansion;
- Generate and test simulated datasets based on alternative and more diverse starting populations and genetic models.

Chromosome 4p15-16

- Use the results presented in this thesis to prioritise the future analysis. This may include resequencing of *significant region 1* and a study of gene expression data;
- Investigate the possibility of gaining access to data from an additional family to be included in the analysis.

References

Abecasis GR, Cherny SS, Cookson WO, Cardon LR (2002) Merlin-rapid analysis of dense genetic maps using sparse gene flow trees. Nat Genet 30:97-101

Allen AS, Satten GA (2007) Statistical models for haplotype sharing in case-parent trio data. Hum Hered. 64(1):35-44. Epub 2007 Apr 27.

Als TD, Dahl HA, Flint TJ, Wang AG, Vang M, Mors O, Kruse TA, Ewald H (2004) Possible evidence for a common risk locus for bipolar affective disorder and schizophrenia on chromosome 4p16 in patients from the Faroe Islands. Mol Psychiatry 9:93-98

American Psychiatric Association (1994) Diagnostic and Statistical Manual of Mental Disorders (Fourth Edition).

Anguelova M, Benkelfat C, Turecki G (2003) A systematic review of association studies investigating genes coding for serotonin receptors and the serotonin transporter: I. Affective disorders. Mol Psychiatry.8:574-91

Asherson P, Mant R, Williams N, Cardno A, Jones L, Murphy K, Collier DA, Nanko S, Craddock N, Morris S, Muir W, Blackwood B, McGuffin P, Owen MJ (1998) A study of chromosome 4p markers and dopamine D5 receptor gene in schizophrenia and bipolar disorder. Mol Psychiatry 3:310-320

Bacanu SA, Devlin B, Roeder K (2000) The power of genomic control. Am J Hum Genet. 66(6):1933-44..

Barden N, Harvey M, Gagné B, Shink E, Tremblay M, Raymond C, Labbé M, Villeneuve A, Rochette D, Bordeleau L, Stadler H, Holsboer F, Müller-Myhsok B (2006) Analysis of single nucleotide polymorphisms in genes in the chromosome 12Q24.31 region points to P2RX7 as a susceptibility gene to bipolar affective disorder. Am J Med Genet B Neuropsychiatr Genet 141(4):374-82

Barrett TB, Hauger RL, Kennedy JL, Sadovnick AD, Remick RA, Keck PE, McElroy SL, Alexander M, Shaw SH, Kelsoe JR (2003) Mol Psychiatry 8:546-57

Barrett JC, Fry B, Maller J, Daly MJ. (2005) Haploview: analysis and visualization of LD and haplotype maps. Bioinformatics 21(2):263-5

Becker T, Knapp M (2004) A powerful strategy to account for multiple testing in the context of haplotype analysis. Am J Hum Genet 75:561-570

Beckmann L, Fischer C, Obreiter M, Rabes M, Chang-Claude J (2005) Haplotype-sharing analysis using Mantel statistics for combined genetic effects. BMC Genet. 6 Suppl 1:S70.

Berrettini WH (2000) Are schizophrenic and bipolar disorders related? A review of family and molecular studies. Biol Psychiatry 48:531-538

Bertelsen A, Harvald B, Hauge M (1977) A Danish twin study of manic-depressive disorders. Br J Psychiatry 130:330-51

Blackwood DH, He L, Morris SW, McLean A, Whitton C, Thomson M, Walker MT, Woodburn K, Sharp CM, Wright AF, Shibasaki Y, St Clair DM, Porteous DJ, Muir WJ (1996) A locus for bipolar affective disorder on chromosome 4p. Nat Genet 12:427-430

Bodmer WF, Bailey CJ, Bodmer J, Bussey HJ, Ellis A, Gorman P, Lucibello FC, Murday VA, Rider SH, Scambler P, et al. (1987) Localization of the gene for familial adenomatous polyposis on chromosome 5. Nature 328(6131):614-6

Botstein D, Risch N (2003) Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease. Nat Genet 33 Suppl:228-37

Bourgain C, Genin E, Quesneville H, Clerget-Darpoux F (2000) Search for multifactorial disease susceptibility genes in founder populations. Ann Hum Genet 64:255-265

Bourgain C, Genin E, Margaritte-Jeannin P, Clerget-Darpoux F (2001) Maximum identity length contrast: a powerful method for susceptibility gene detection in isolated populations. Genet Epidemiol 21 Suppl 1:S560-4

Bourgain C, Genin E, Ober C, Clerget-Darpoux F (2002) Missing data in haplotype analysis: a study on the AS method. Ann Hum Genet 66(Pt 1):99-108

Chakravarti A (1999) Population genetics – making sense out of sequence. Nat Genet Suppl 21:56:60

Chapman NH, Thompson EA (2001) Linkage disequilibrium mapping: the role of population history, size, and structure. Adv Genet 42:413-37

Chen YS, Akula N, Detera-Wadleigh SD, Schulze TG, Thomas J, Potash JB, DePaulo JR, McInnis MG, Cox NJ, McMahon FJ (2004) Findings in an independent sample support an association between bipolar affective disorder and the G72/G30 locus on chromosome 13q33. Mol Psychiatry 9(1):87-92

Christoforou A, Le Hellard S, Thomson P, Morris S, Tenesa A, Pickard B. S, Wray NR, Muir W, Blackwood D, Porteous DJ et al. (2007) Association Analysis of the Chromosome 4p15-p16 Candidate Region for Bipolar Disorder and Schizophrenia. Mol Psychiatry [Epub ahead of print]

Clayton D (1999) A generalization of the transmission/disequilibrium test for uncertain-haplotype transmission. Am J Hum Genet 65(4):1170-7

Clayton D (2000) Linkage disequilibrium mapping of disease susceptibility genes in human populations. International Statistical Review, 68:23-43

Cohen JC, Kiss RS, Pertsemlidis A, Marcel YL, McPherson R, Hobbs HH (2004) Multiple rare alleles contribute to low plasma levels of HDL cholesterol. Science 6;305(5685):869-72

Collins FS (1995) Positional cloning moves from perditional to traditional. Nat Genet 9(4):347-50

Corder EH, Saunders AM, Strittmatter WJ, Schmechel DE, Gaskell PC, Small GW, Roses AD, Haines JL, Pericak-Vance MA (1993) Gene dose of apolipoprotein E type 4 allele and the risk of Alzheimer's disease in late onset families. Science 261(5123):921-3

Craddock N, Forty L (2006) Genetics of affective (mood) disorders. Eur J Hum Genet 14(6):660-8

Craddock N, Khodel V, Van Eerdewegh P, Reich T (1995) Mathematical limits of multilocus models: the genetic transmission of bipolar disorder. Am J Hum Genet 57: 690-702

Craddock N, Jones I (1999) Genetics of bipolar disorder. J Med Genet 36(8):585-94

Culverhouse R, Lin J, Liu KY, Suarez BK (2001) Exploiting linkage disequilibrium in population isolates. Genet Epidemiol 21 Suppl 1:S429-34

Das Gupta R, Guest JF (2002) Annual cost of bipolar disorder to UK society. Br J Psychiatry 180:227-33

Davies JL, Kawaguchi Y, Bennett ST, Copeman JB, Cordell HJ, Pritchard LE, Reed PW, Gough SC, Jenkins SC, Palmer SM, et al. (1994) A genome-wide search for human type 1 diabetes susceptibility genes. Nature 371(6493):130-6

De Braekeleer M, Chaventré A, Bertorelle G, Verlingue C, Raguénès O, Mercier B, Férec C (1996) Linkage disequilibrium between the four most common cystic fibrosis mutations and microsatellite haplotypes in the Celtic population of Brittany. Hum Genet 98(2):223-7

de la Chapelle A, Wright FA (1998) Linkage disequilibrium mapping in isolated populations: the example of Finland revisited. Proc Natl Acad Sci U S A 13;95(21):12416-23

Detera-Wadleigh SD, Badner JA, Berrettini WH, Yoshikawa T, Goldin LR, Turner G, Rollins DY, Moses T, Sanders AR, Karkera JD, Esterling LE, Zeng J, Ferraro TN, Guroff JJ, Kazuba D, Maxwell ME, Nurnberger JI, Jr., Gershon ES (1999) A high-density genome scan detects evidence for a bipolar-disorder susceptibility locus on 13q32 and other potential loci on 1q32 and 18p11.2. Proc Natl Acad Sci U S A 96:5604-5609

Devlin B, Roeder K (1999) Genomic control for association studies. Biometrics. 55(4):997-1004.

Diabetes Genetics Initiative of Broad Institute of Harvard and MIT, Lund University, and Novartis Institutes of BioMedical Research, Saxena R, Voight BF, Lyssenko V, Burtt NP, de Bakker PI, Chen H, Roix JJ, Kathiresan S, Hirschhorn JN et al. (2007) Genome-wide association analysis identifies loci for type 2 diabetes and triglyceride levels. Science 316(5829):1331-6

Dreyer SD, Zhou G, Baldini A, Winterpacht A, Zabel B, Cole W, Johnson RL, Lee B (1998) Mutations in LMX1B cause abnormal skeletal patterning and renal dysplasia in nail patella syndrome. Nat Genet 19(1):47-50

Duggal P, Klein AP, Lee KE, Klein R, Klein BEK, Bailey-Wilson JE (2007) Identification of Novel Genetic Loci for Intraocular Pressure: A Genomewide Scan of the Beaver Dam Eye Study. Arch Ophthalmol 125(1):74-79

Enattah NS, Sahi T, Savilahti E, Terwilliger JD, Peltonen L, Järvelä I (2002) Identification of a variant associated with adult-type hypolactasia. Nat Genet 30(2):233-7

Ewald H, Degn B, Mors O, Kruse TA (1998) Support for the possible locus on chromosome 4p16 for bipolar affective disorder. Mol Psychiatry 3:442-448

Fallin D, Schork NJ (2000) Accuracy of haplotype frequency estimation for biallelic loci, via the expectation-maximization algorithm for unphased diploid genotype data. Am J Hum Genet 67(4):947-59. Epub

Fan R, Lange K (1998) Models for haplotype evolution in a nonstationary population. Theor Popul Biol 53(3):184-98

Faravelli C, Guerrini Degl'Innocenti B, Aiazzi L, Incerpi G, Pallanti S (1990) Epidemiology of mood disorders: a community survey in Florence. J Affect Disord 20(2):135-41

Feder JN, Gnirke A, Thomas W, Tsuchihashi Z, Ruddy DA, Basava A, Dormishian F, Domingo R Jr, Ellis MC, Fullan A, Hinton LM, Jones NL, Kimmel BE, Kronmal GS, Lauer P, Lee VK, Loeb DB, Mapa FA, McClelland E, Meyer NC, Mintier GA, Moeller N, Moore T, Morikang E, Prass CE, Quintana L, Starnes SM, Schatzman RC, Brunke KJ, Drayna DT, Risch NJ, Bacon BR, Wolff RK (1996) A novel MHC class I-like gene is mutated in patients with hereditary haemochromatosis. Nat Genet 13(4):399-408

Feuk L, Marshall CR, Wintle RF, Scherer SW (2006) Structural variants: changing the landscape of chromosomes and design of disease studies. Hum Mol Genet 15 Spec No 1:R57-66

Ge Y, Dudoit S, Speed TP (2003) Resampling-based multiple testing for microarray data analysis. Technical Report 633, Department of Statistics, University of California, Berkeley

Geller B, Badner JA, Tillman R, Christian SL, Bolhofner K, Cook EH Jr (2004) Linkage disequilibrium of the brain-derived neurotrophic factor Val66Met polymorphism in children with a prepubertal and early adolescent bipolar disorder phenotype. Am J Psychiatry 161(9):1698-700

Goldney RD, Fisher LJ, Grande ED, Taylor AW, Hawthorne G (2005) Bipolar I and II disorders in a random and representative Australian population. Aust N Z J Psychiatry 39(8):726-9

Hall JM, Lee MK, Newman B, Morrow JE, Anderson LA, Huey B, King MC (1990) Linkage of early-onset familial breast cancer to chromosome 17q21. Science 250(4988):1684-9

Hattori E, Liu C, Badner JA, Bonner TI, Christian SL, Maheshwari M, Detera-Wadleigh SD, Gibbs RA, Gershon ES (2003) Polymorphisms at the G72/G30 gene locus, on 13q33, are associated with bipolar disorder in two independent pedigree series. Am J Hum Genet 72(5):1131-40. Epub 2003 Mar 19

Hedrick PW (1987) Gametic disequilibrium measures: proceed with caution. Genetics 117: 331-341

Herzberg I, Jasinska A, García J, Jawaheer D, Service S, Kremeyer B, Duque C, Parra MV, Vega J, Ortiz D et al. (2006) Convergent linkage evidence from two Latin-American population isolates supports the presence of a susceptibility locus for bipolar disorder in 5q31-34. Hum Mol Genet 15(21):3146-53

Hong CJ, Huo SJ, Yen FC et al. (2003) Association study of a brainderived neurotrophic-factor genetic polymorphism and mood disorders, age of onset and suicidal behaviour. Neuropsychobiology 48: 186-189

Hu F, Post J, Johnson S, Ehrlich G, Preston R (2000) Refined localization of a gene for pediatric gastroesophageal reflux makes HTR2A an unlikely candidate gene. Hum Genet 107(5):519-525

Hubbard TJ, Aken BL, Beal K, Ballester B, Caccamo M, Chen Y, Clarke L, Coates G, Cunningham F, Cutts T, Down T, Dyer SC, Fitzgerald S, Fernandez-Banet J, Graf S, Haider S, Hammond M, Herrero J, Holland R, Howe K, Howe K, Johnson N, Kahari A, Keefe D, Kokocinski F, Kulesha E, Lawson D, Longden I, Melsopp C, Megy K, Meidl P, Ouverdin B, Parker A, Prlic A, Rice S, Rios D, Schuster M, Sealy I, Severin J, Slater G, Smedley D, Spudich G, Trevanion S, Vilella A, Vogel J, White S, Wood M, Cox T, Curwen V, Durbin R, Fernandez-Suarez XM, Flicek P, Kasprzyk A, Proctor G, Searle S, Smith J, Ureta-Vidal A, Birney E (2007) Ensembl 2007. Nucleic Acids Res 35(Database issue):D610-7

Hunter DJ, Kraft P (2007) Drinking from the fire hose--statistical issues in genomewide association studies. N Engl J Med 357(5):443-53

Huntington's Disease Collaborative Research Group (1993) A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease

chromosomes. The Huntington's Disease Collaborative Research Group. Cell 72(6):971-83

Iafrate AJ, Feuk L, Rivera MN, Listewnik ML, Donahoe PK, Qi Y, Scherer SW, Lee C (2004) Detection of large-scale variation in the human genome. Nat Genet 36(9):949-51

Jones I, Craddock N (2001) Candidate gene studies of bipolar disorder. Ann Med 33(4):248-56

Jorde LB (2000) Linkage disequilibrium and the search for complex disease genes. Genome Res 10(10):1435-44

Kakiuchi C, Iwamoto K, Ishiwata M, Bundo M, Kasahara T, Kusumi I, Tsujita T, Okazaki Y, Nanko S, Kunugi H, Sasaki T, Kato T (2003) Impaired feedback regulation of XBP1 as a genetic risk factor for bipolar disorder. Nat Genet 35(2):171-5

Ke X, Durrant C, Morris AP, Hunt S, Bentley DR, Deloukas P, Cardon LR (2004) Efficiency and consistency of haplotype tagging of dense SNP maps in multiple samples. Hum Mol Genet 13: 2557-2565

Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D (2002) The human genome browser at UCSC. Genome Res 12(6):996-1006

Kerem B, Rommens JM, Buchanan JA, Markiewicz D, Cox TK, Chakravarti A, Buchwald M, Tsui LC (1989) Identification of the cystic fibrosis gene: genetic analysis Science 245(4922):1073-80

Klei L, Roeder K (2007) Testing for association based on excess allele sharing in a sample of related cases and controls. Hum Genet 121(5): 549-557

Koenig M, Hoffman EP, Bertelson CJ, Monaco AP, Feener C, Kunkel LM (1986) Complete cloning of the Duchenne muscular dystrophy (DMD) cDNA and preliminary genomic organization of the DMD gene in normal and affected individuals. Cell 50(3):509-17

Koskenmies S, Widén E, Onkamo P, Sevón P, Julkunen H, Kere J. (2004) Haplotype associations define target regions for susceptibility loci in systemic lupus erythematosus. Eur Jour Hum Genet 12(6): 489-494

Lachman HM, Pedrosa E, Petruolo OA, Cockerham M, Papolos A, Novak T, Papolos DF, Stopkova P (2007) Increase in GSK3beta gene copy number variation in bipolar disorder. Am J Med Genet B Neuropsychiatr Genet. 144B(3):259-65.

Lander E, Kruglyak L (1995) Genetic dissection of complex traits: guidelines for interpreting and reporting linkage results. Nat Genet 11(3):241-7

Lander ES, Schork NJ (1994) Genetic dissection of complex traits. Science. 265(5181):2037-48.

Lasky-Su JA, Faraone SV, Glatt SJ, Tsuang MT (2005) Meta-analysis of the association between two polymorphisms in the serotonin transporter gene and affective disorders. Am J Med Genet B Neuropsychiatr Genet 133(1):110-5

Le Hellard S, Lee AJ, Underwood S, Thomson PA, Morris SW, Torrance HS, Anderson SM, Adams RR, Navarro P, Christoforou A et al. (2006) Haplotype Analysis and a Novel Allele-sharing Method Refines a Chromosome 4p Locus Linked to Bipolar Affective Disorder. Biol Psychiatry 61(6):797-805

Lerer B, Segman RH, Hamdan A, Kanyas K, Karni O, Kohn Y, Korner M, Lanktree M, Kaadan M, Turetsky N, Yakir A, Kerem B, Macciardi F (2003) Genome scan of Arab Israeli families maps a schizophrenia susceptibility gene to chromosome 6q23 and supports a locus at chromosome 10q24. Mol Psychiatry 8:488-498

Levinson DF, Levinson MD, Segurado R, Lewis CM. (2003) Genome scan meta-analysis of schizophrenia and bipolar disorder, part I: Methods and power analysis. Am J Hum Genet 73(1):17-33

Lopez AD, Murray CC (1998) The global burden of disease, 1990-2020. Nat Med 4(11):1241-3

Lunetta KL, Faraone SV, Biederman J, Laird NM. (2000) Family-based tests of association and linkage that use unaffected sibs, covariates, and interactions. Am J Hum Genet. 66(2):605-14.

Macgregor S, Visscher PM, Knott SA, Thomson P, Porteous DJ, Millar JK, Devon RS, Blackwood D, Muir WJ (2004) A genome scan and follow-up study identify a bipolar disorder susceptibility locus on chromosome 1q42. Mol Psychiatry 9(12):1083-90

Mantel, N. 1967. The detection of disease clustering and a generalized regression approach. Cancer Res 27:209-220.

Marchini J, Cardon LR, Phillips MS, Donnelly P (2004) The effects of human population structure on large genetic association studies. Nat Genet. 36(5):512-7.

Martin ER, Monks SA, Warren LL, Kaplan NL (2000) A test for linkage and association in general pedigrees: the pedigree disequilibrium test. Am J Hum Genet. 67(1):146-54.

McKusick VA Mendelian Inheritance in Man. A Catalog of Human Genes and Genetic Disorders (12th edition) 1998 John Hopkins University Press, Baltimore, Md. USA

Miki Y, Swensen J, Shattuck-Eidens D, Futreal PA, Harshman K, Tavtigian S, Liu Q, Cochran C, Bennett LM, Ding W, et al. (1994) A strong candidate for the breast and ovarian cancer susceptibility gene BRCA1. Science 266(5182):66-71

Miyazawa H, Kato M, Awata T, Kohda M, Iwasa H, Koyama N, Tanaka T, Huqun , Kyo S, Okazaki Y, Hagiwara K. (2007) Homozygosity haplotype allows a genomewide search for the autosomal segments shared among patients. Am J Hum Genet 80(6):1090-102

Molitor J, Marjoram P, Thomas D (2003) Application of Bayesian spatial statistical methods to analysis of haplotypes effects and gene mapping. Genet Epidemiol 25(2): 95-105

Molitor J, Marjoram P, Thomas D (2003) Fine-scale mapping of disease genes with multiple mutations via spatial clustering techniques. Am J Hum Genet 73(6): 1368-1384

Morissette J, Villeneuve A, Bordeleau L, Rochette D, Laberge C, Gagné B, Laprise C, Bouchard G, Plante M, Gobeil L et al. (1999) Genome-wide search for linkage of bipolar affective disorders in a very large pedigree derived from a homogeneous population in quebec points to a locus of major effect on chromosome 12q23-q24. Am J Med Genet 88(5):567-87

Morris AP (2005) Direct analysis of unphased SNP genotype data in population-based association studies via Bayesian partition modelling of haplotypes. Genet Epidemiol 29(2): 91-107

Morris AP (2006) A flexible Bayesian framework for modeling haplotype association with disease, allowing for dominance effects of the underlying causative variants. Am J Hum Genet (4): 679-94

Morton NE, Collins A (1998) Tests and estimates of allelic association in complex inheritance. Proc Natl Acad Sci USA. 95(19):11389-93.

Nachman NW (2002) Variation in recombination rate across the genome: evidence and implications. Curr Opin Genet Dev 12(6):657-63

Nakata K, Ujike H, Sakai A, Uchida N, Nomura A, Imamura T, Katsu T, Tanaka Y, Hamamura T, Kuroda S (2003) Association study of the brain-derived neurotrophic factor (BDNF) gene with bipolar disorder. Neurosci Lett 337(1):17-20

NCI-NHGRI Working Group on Replication in Association Studies, Chanock SJ, Manolio T, Boehnke M, Boerwinkle E, Hunter DJ, Thomas G, Hirschhorn JN, Abecasis G, Altshuler D, Bailey-Wilson JE, Brooks LD, Cardon LR, Daly M, Donnelly P, Fraumeni JF Jr, Freimer NB, Gerhard DS, Gunter C, Guttmacher AE, Guyer MS, Harris EL, Hoh J, Hoover R, Kong CA, Merikangas KR, Morton CC, Palmer LJ, Phimister EG, Rice JP, Roberts J, Rotimi C, Tucker MA, Vogan KJ, Wacholder S, Wijsman EM, Winn DM, Collins FS (2007) Replicating genotype-phenotype associations. Nature 447(7145):655-660

Neves-Pereira M, Mundo E, Muglia P, King N, Macciardi F, Kennedy JL (2002) The brain-derived neurotrophic factor gene confers susceptibility to bipolar disorder: evidence from a family-based association study. Am J Hum Genet 71(3):651-5

Nishisho I, Nakamura Y, Miyoshi Y, Miki Y, Ando H, Horii A, Koyama K, Utsunomiya J, Baba S, Hedge P (1991) Mutations of chromosome 5q21 genes in FAP and colorectal cancer patients. Science 253(5020):665-9

Oswald P, Del-Favero J, Massat I, Souery D, Claes S, Van Broeckhoven C, Mendlewicz J (2004) Non-replication of the brain-derived neurotrophic factor (BDNF) association in bipolar affective disorder: a Belgian patient-control study. Am J Med Genet B Neuropsychiatr Genet 129(1):34-5

Ott J (1999) Methods of analysis and resources available for genetic trait mapping. J Hered 90(1):68-70

Pearson TA, Manolio TA 2008 How to interpret a genome-wide association study. JAMA 299(11):1335-44
Pini S, de Queiroz V, Pagnin D, Pezawas L, Angst J, Cassano GB, Wittchen HU (2005) Prevalence and burden of bipolar disorders in European countries. Eur Neuropsychopharmacol 15(4):425-34

Preisig M, Bellivier F, Fenton BT, Baud P, Berney A, Courtet P, Hardy P, Golaz J, Leboyer M, Mallet J, Matthey ML, Mouthon D, Neidhart E, Nosten-Bertrand M, Stadelmann-Dubuis E, Guimon J, Ferrero F, Buresi C, Malafosse A (2000) Association between bipolar disorder and monoamine oxidase A gene polymorphisms: results of a multicenter study. Am J Psychiatry 157(6):948-55

Pritchard JK, Rosenberg NA (1999) Use of unlinked genetic markers to detect population stratification in association studies. Am J Hum Genet. 65(1):220-8.

Pritchard JK, Stephens M, Rosenberg NA, Donnelly P (2000) Association mapping in structured populations. Am J Hum Genet. 67(1):170-81.

Qian D, Thomas DC (2001) Genome scan of complex traits by haplotype sharing correlation. Genet Epidemiol. 21 Suppl 1:S582-7.

Regeer EJ, ten Have M, Rosso ML, Hakkaart-van Roijen L, Vollebergh W, Nolen WA (2004) Prevalence of bipolar disorder in the general population: a Reappraisal Study of the Netherlands Mental Health Survey and Incidence Study. Acta Psychiatr Scand 110(5):374-82

Regier DA, Farmer ME, Rae DS, Locke BZ, Keith SJ, Judd LL, Goodwin FK (1990) Comorbidity of mental disorders with alcohol and other drug abuse. Results from the Epidemiologic Catchment Area (ECA) Study. JAMA 264(19):2511-8

Riordan JR, Rommens JM, Kerem B, Alon N, Rozmahel R, Grzelczak Z, Zielenski J, Lok S, Plavsic N, Chou JL, Drumm ML, Iannuzzi MC, Collins FS, Tsui LC (1989) Identification of the cystic fibrosis gene: cloning and characterization of complementary DNA. Science 245(4922):1066-73

Royer-Pokora B, Kunkel LM, Monaco AP, Goff SC, Newburger PE, Baehner RL, Cole FS, Curnutte JT, Orkin SH (1986) Cloning the gene for an inherited human disorder--chronic granulomatous disease--on the basis of its chromosomal location.

Nature 322(6074):32-8

Savitsky K, Bar-Shira A, Gilad S, Rotman G, Ziv Y, Vanagaite L, Tagle DA, Smith S, Uziel T, Sfez S, Ashkenazi M, Pecker I, Frydman M, Harnik R, Patanjali SR, Simmons A, Clines GA, Sartiel A, Gatti RA, Chessa L, Sanal O, Lavin MF, Jaspers NG, Taylor AM, Arlett CF, Miki T, Weissman SM, Lovett M, Collins FS, Shiloh Y (1995) A single ataxia telangiectasia gene with a product similar to PI-3 kinase. Science 268(5218):1749-53

Schumacher J, Jamra RA, Freudenberg J, Becker T, Ohlraun S, Otte AC, Tullius M, Kovalenko S, Bogaert AV, Maier W, Rietschel M, Propping P, Nöthen MM, Cichon S (2004) Examination of G72 and D-amino-acid oxidase as genetic risk factors for schizophrenia and bipolar affective disorder. Mol Psychiatry 9(2):203-7

Scotet V, Gillet D, Dugueperoux I, Audrezet MP, Bellis G, Garnier B, Roussey M, Rault G, Parent P, De Braekeleer M et al. (2002) Spatial and temporal distribution of cystic fibrosis and of its mutations in Brittany, France: a retrospective study from 1960. Hum Genet 111(3):247-54

Sebat J, Lakshmi B, Troge J, Alexander J, Young J, Lundin P, Månér S, Massa H, Walker M, Chi M, Navin N, Lucito R, Healy J, Hicks J, Ye K, Reiner A, Gilliam TC, Trask B, Patterson N, Zetterberg A, Wigler M (2004) Large-scale copy number polymorphism in the human genome. Science 305(5683):525-8

Sebat J, Lakshmi B, Malhotra D, Troge J, Lese-Martin C, Walsh T, Yamrom B, Yoon S, Krasnitz A, Kendall J, Leotta A, Pai D, Zhang R, Lee YH, Hicks J, Spence SJ, Lee AT, Puura K, Lehtimäki T, Ledbetter D, Gregersen PK, Bregman J, Sutcliffe JS, Jobanputra V, Chung W, Warburton D, King MC, Skuse D, Geschwind DH, Gilliam TC, Ye K, Wigler M (2007) Strong association of de novo copy number mutations with autism. Science 316(5823):445-9

Skibinska M, Hauser J, Czerski PM, Leszczynska-Rodziewicz A, Kosmowska M, Kapelski P, Slopien A, Zakrzewska M, Rybakowski JK (2004) Association analysis of brain-derived neurotrophic factor (BDNF) gene Val66Met polymorphism in schizophrenia and bipolar affective disorder. World J Biol Psychiatry 5(4):215-20

Sklar P, Gabriel SB, McInnis MG, Bennett P, Lim YM, Tsan G, Schaffner S, Kirov G, Jones I, Owen M, Craddock N, DePaulo JR, Lander ES (2002) Family-based association study of 76 candidate genes in bipolar disorder: BDNF is a potential risk locus. Brain-derived neutrophic factor. Mol Psychiatry 7(6):579-93

Sklar P, Smoller JW, Fan J, Ferreira MA, Perlis RH, Chambert K, Nimgaonkar VL, McQueen MB, Faraone SV, Kirby A, de Bakker PI, Ogdie MN, Thase ME, Sachs GS, Todd-Brown K, Gabriel SB, Sougnez C, Gates C, Blumenstiel B, Defelice M, Ardlie KG, Franklin J, Muir WJ, McGhee KA, MacIntyre DJ, McLean A, VanBeck M, McQuillin A, Bass NJ, Robinson M, Lawrence J, Anjorin A, Curtis D, Scolnick EM, Daly MJ, Blackwood DH, Gurling HM, Purcell SM (2008) Whole-genome association study of bipolar disorder. Mol Psychiatry. 13(6):558-69.

Slooter AJ, van Duijn CM (1997) Genetic epidemiology of Alzheimer disease. Epidemiol Rev 19(1):107-19

Spielman RS, McGinnis RE, Ewens WJ (1993) Transmission test for linkage disequilibrium: the insulin gene region and insulin-dependent diabetes mellitus (IDDM). Am J Hum Genet. 52(3):506-16.

Stefansson H, Sigurdsson E, Steinthorsdottir V, Bjornsdottir S, Sigmundsson T, Ghosh S, Brynjolfsson J, Gunnarsdottir S, Ivarsson O, Chou TT, Hjaltason O, Birgisdottir B, Jonsson H, Gudnadottir VG, Gudmundsdottir E, Bjornsson A, Ingvarsson B, Ingason A, Sigfusson S, Hardardottir H, Harvey RP, Lai D, Zhou M, Brunner D, Mutel V, Gonzalo A, Lemke G, Sainz J, Johannesson G, Andresson T, Gudbjartsson D, Manolescu A, Frigge ML, Gurney ME, Kong A, Gulcher JR, Petursson H, Stefansson K (2002) Neuregulin 1 and susceptibility to schizophrenia. Am J Hum Genet. 71(4):877-92.

Strathdee CA, Gavish H, Shannon WR, Buchwald M (1992) Cloning of cDNAs for Fanconi's anaemia by functional complementation. Nature 356(6372):763-7

Szádóczky E, Papp Z, Vitrai J, Ríhmer Z, Füredi J (1998) The prevalence of major depressive and bipolar disorders in Hungary. Results from a national epidemiologic survey. J Affect Disord 50(2-3):153-62

te Meerman GJ, Van der Meulen MA (1997) Genomic sharing surrounding alleles identical by descent: effects of genetic drift and population growth. Genet Epidemiol 14(6): 1125-1130

ten Have M, Vollebergh W, Bijl R, Nolen WA (2002) Bipolar disorder in the general population in The Netherlands (prevalence, consequences and care utilisation): results from The Netherlands Mental Health Survey and Incidence Study (NEMESIS). J Affect Disord 68(2-3):203-13

The International HapMap Consortium (2003) The International HapMap Project. Nature 426, 789-796

Thierry-Mieg D, Thierry-Mieg J (2006) AceView: a comprehensive cDNA-supported gene and transcripts annotation. Genome Biol, 7 (Suppl 1):S12

Thomson PA, Wray NR, Millar JK, Evans KL, Hellard SL, Condie A, Muir WJ, Blackwood DH, Porteous DJ (2005) Association between the TRAX/DISC locus and both bipolar disorder and schizophrenia in the Scottish population. Mol Psychiatry 10: 657-668, 616

Tomlinson I, Webb E, Carvajal-Carmona L, Broderick P, Kemp Z, Spain S, Penegar S, Chandler I, Gorman M, Wood W et al. (2007) A genome-wide association scan of tag SNPs identifies a susceptibility variant for colorectal cancer at 8q24.21. Nat Genet 39(8):984-988

Twyman R (2003) Finding the rough position of human disease genes relative to known genetic markers. http://genome.wellcome.ac.uk/doc_wtd020778.html

Tzeng JY, Devlin B, Wasserman L, Roeder K (2003) On the identification of disease mutations by the analysis of haplotype similarity and goodness of fit. Am J Hum Genet 72:891-902

Tzeng, J.Y., Byerley, W., Devlin, B., Roeder, K. and Wasserman, L. (2003). Outlier detection and false discovery rates for whole-genome DNA matching. Journal of the American Statistical Association, 98:236-246.

Tzeng JY, Wang CH, Kao JT, Hsiao CK (2006) Regression-based association analysis with clustered haplotypes through use of genotypes. Am J Hum Genet 78(2): 231-242

Tzeng JY, Zhang D (2007) Haplotype-based association analysis via variance-components score test. Am J Hum Genet 81(5): 927-938

Vallès V, Van Os J, Guillamat R, Gutiérrez B, Campillo M, Gento P, Fañanás L (2000) Increased morbid risk for schizophrenia in families of in-patients with bipolar illness. Schizophr Res 42(2):83-90

Van der Meulen MA, te Meerman GJ (1997) Haplotype sharing analysis in affected individuals from nuclear families with at least one affected offspring. Genet Epidemiol 14:915-920

Venken T, Alaerts M, Souery D, Goossens D, Sluijs S, Navon R, Van Broeckhoven C, Mendlewicz J, Del-Favero J, Claes S (2008) Chromosome 10q harbors a susceptibility locus for bipolar disorder in Ashkenazi Jewish families. Mol Psychiatry 13(4):442-50

Visscher PM, Haley CS, Heath SC, Muir WJ, Blackwood DH (1999) Detecting QTLs for uni- and bipolar disorder using a variance component method. Psychiatr Genet 9:75-84

Voight BF, Pritchard JK (2005) Confounding from cryptic relatedness in case-control association studies. PLoS Genet. 1(3):e32.

Wallace MR, Marchuk DA, Andersen LB, Letcher R, Odeh HM, Saulino AM, Fountain JW, Brereton A, Nicholson J, Mitchell AL, Brownstein BH, Collins FS (1990) Type 1 neurofibromatosis gene: identification of a large transcript disrupted in three NF1 patients. Science 249(4965):181-6

Weinberg CR (1999) Methods for detection of parent-of-origin effects in genetic studies of case-parents triads. Am J Hum Genet. 65(1):229-35.

Weiss KM, Clark AG (2002) Linkage disequilibrium and the mapping of complex human traits. Trends Genet 18:19-24

Weissman MM, Bland RC, Canino GJ, Faravelli C, Greenwald S, Hwu HG, Joyce PR, Karam EG, Lee CK, Lellouch J, Lépine JP, Newman SC, Rubio-Stipec M, Wells

JE, Wickramaratne PJ, Wittchen H, Yeh EK (1996) Cross-national epidemiology of major depression and bipolar disorder. JAMA 276(4):293-9

Wiesner GL, Daley D, Lewis S, Ticknor C, Platzer P, Lutterbaugh J, MacMillen M, Baliner B, Willis J, Elston RC, Markowitz SD (2003) A subset of familial colorectal neoplasia kindreds linked to chromosome 9q22.2-31.2. Proc Natl Acad Sci USA. 100(22):12961-5

Williams NM, Rees MI, Holmans P, Norton N, Cardno AG, Jones LA, Murphy KC, Sanders RD, McCarthy G, Gray MY, Fenton I, McGuffin P, Owen MJ (1999) A two-stage genome scan for schizophrenia susceptibility genes in 196 affected sibling pairs. Hum Mol Genet 8:1729-1739

Williams NM, Green EK, Macgregor S, Dwyer S, Norton N, Williams H, Raybould R, Grozeva D, Hamshere M, Zammit S, Jones L, Cardno A, Kirov G, Jones I, O'Donovan MC, Owen MJ, Craddock N (2006) Variation at the DAOA/G30 locus influences susceptibility to major mood episodes but not psychosis in schizophrenia and bipolar disorder. Arch Gen Psychiatry 63(4):366-73

Wellcome Trust Case Control Consortium (2007) Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. Nature 447(7145):661-78

Wooster R, Bignell G, Lancaster J, Swift S, Seal S, Mangion J, Collins N, Gregory S, Gumbs C, Micklem G (1995) Identification of the breast cancer susceptibility gene BRCA2. Nature 378(6559):789-92

Appendix A: Key programs written during this thesis

A.1 ASCalculate

```
import java.io.*;
import java.util.Random;
import java.util.Vector;
import java.util.StringTokenizer;
import java.lang.Thread;

/**
 *   Main class
 */
public class ASCalculate{

  private int progCount;
  private String inputPath;
  private int nocases;
  private int nocontrols;
  //ASType - use to determine whether /no of markers/ or /marker dist/ is used as
  //measure of allele sharing
  private int ASType;//0 for no of markers dist, 1 for bp dist
  //0 for ingnore missing data, compare ambigous/1 for ignore all missing and
  //ambiguous
  private int missingType;
  //attributes relating to permutation analysis
  private boolean permAnalysis;
  private String permType;
  private int noPerms;

  Vector pStats_ = new Vector();

  /**
  *  1. The ASCalculate class initiates the variable only
  */
  public ASCalculate(String st, int nocases, int nocontrols, int ASType, int
  missingType, boolean permAnalysis, String permType ,int noPerms) {
      inputPath = st;
      this.nocases = nocases;
      this.nocontrols = nocontrols;
      this.ASType = ASType;
      this.missingType = missingType;
      this.permAnalysis = permAnalysis;
      this.permType = permType;
      this.noPerms = noPerms;
      progCount = 0;
  }

  /**
   *  2. This method is called to actually begin the procedure
   **/
  public void jbInit() throws Exception {

    //******
    //input the data from designated file
    File f = new File(inputPath);
    //create a vector of vectors, one for each case and control column of data
    Vector inputData = new Vector();
    for (int i=0;i<(nocontrols+nocases);i++){
      Vector vect_temp = new Vector();
      inputData.add(vect_temp);
    }
    //create bufered reader to read in the data from file
    BufferedReader br = new BufferedReader(new FileReader(inputPath));
    String line = br.readLine();
    StringTokenizer tok = new StringTokenizer(line,",");
    int noTokens = tok.countTokens();
    Vector[] outputData;//hold results of basic analysis
    //hold results of permutation analysis should it be applicable
    Vector[] permOut = new Vector[0];
    Vector pStats = new Vector();//hold stats results of permutation analysis
```

```
if (noTokens != (nocontrols+nocases)){//there is an error
  Vector v = new Vector();
  outputData = new Vector[1];
  outputData[0] = v;
} else {
  while (line != null){//for each line at a time
    //tokenize each row, and put into the different vectors
    int i=0;
    tok = new StringTokenizer(line,",");
    while(tok.hasMoreTokens()){
      String st = tok.nextToken();
      if (missingType == 0)//take all data as a string for now
        ((Vector)inputData.get(i)).add(st);
      else {//convert data to doubles, where data is not a double, take as '-1'
        if(isInteger(st))
          ((Vector)inputData.get(i)).add(new Double(st));
        else
           //use -1 to recognise a missing or undefined genotype
          ((Vector)inputData.get(i)).add(new Double(-1.0));
      }
      i++;
    }
    //get new line and loop
    line = br.readLine();
  }
  br.close();

  //*****************************
  //3. implement the AS algorithm
  //return two vectors
  //now, use the  method to take the average of both sets of pairwise comparisons
  outputData = new Vector[3];
  //get average of cases pairwise AS and
  outputData[0] = average(calcAS(inputData,0,nocases));
  //get average of ctrls pairwise AS
  outputData[1] = average(calcAS(inputData,nocases,(nocases+nocontrols)));
  //get the difference between the two averages
  outputData[2] = difference(outputData[0],outputData[1]);

  //*************************************
  //4. implement PermAnalysis (if requested)
  //return Vector[3]
  if(permAnalysis == true){//if we choose to carry out permutation analysis...
    Vector permTempTemp = new Vector();
    for (int n=0;n<diseaseHap.length;n++)
      permTempTemp.add(new Double(diseaseHap[n]));
    String outputPath = f.getParent() + "\\permutations.log";
    BufferedWriter bwPermLog = new BufferedWriter(new FileWriter(outputPath));
    bwPermLog.newLine();
    bwPermLog.write("Permutations of input chromosomes are as follows:");
    if(permType == "standard")
      permOut = new Vector[noPerms];
    else if (permType == "modified")
      permOut = new Vector[noPerms];
    for(int i=0;i<noPerms;i++){
      //get random nos
      bwPermLog.newLine();
      //temp vector to hold the randomised input columns
      Vector permTempIn = new Vector();
      Random rand = new Random(System.currentTimeMillis());
      //to make sure there is a unique seed for the random number generator
      Thread.sleep(10);
      Vector randList = new Vector();
      if(permType == "standard"){
        for (int j=0;j<inputData.size();j++){
          int randNo = getRnd(inputData.size(),rand);
          //if we have already taken that chr
          if (randList.contains(new Integer(randNo))){
            j--;//repeat loop
          } else {
            //write the chr taken to log
            bwPermLog.write( (new Integer(randNo)).toString() );
            bwPermLog.write(",");//write the chr taken to log
            permTempIn.add(inputData.get(randNo));//add chr to permTempIn
            //add to vector so it doesnt get taken again
            randList.add(new Integer(randNo));
          }
```

```
        }//end of get random nos
        //use progCount to keep track of the number of permutations that have
        //been carried out
        progCount = i;
        permOut[i]=difference(average(calcAS(permTempIn,0,nocases)),
        average(calcAS(permTempIn,nocases,(nocases+nocontrols))) );
      }
      } else if (permType == "modified"){//MODIFIED PERMUTATION ANALYSIS
        //first add the known disease hap to permTempIn
        permTempIn.add(permTempTemp);
        //now find (nocases-1) random haplotypes and add them to permTempIn
        for (int j=0;j<nocases-1;j++){
          int randNo = getRnd(inputData.size(),rand);
          //if we have already taken that chr
          if (randList.contains(new Integer(randNo))){
            j--;//repeat loop
          } else {
            //write the chr taken to log
            bwPermLog.write( (new Integer(randNo)).toString() );
            bwPermLog.write(",");//write the chr taken to log
            permTempIn.add(inputData.get(randNo));//add chr to permTempIn
            //add to vector so it doesnt get taken again
            randList.add(new Integer(randNo));
          }
        }//end of get random nos
        //in the modified version, only want to get the average of cases
        permOut[i] = average(calcAS(permTempIn,0,nocases));
      }//end of if modified
    }//end of for(noPerms)
    //close the log file
    bwPermLog.close();
    //calc permutation statistics
    if (permType == "standard"){
      pStats = permStats(permOut, outputData);
    }
    else if (permType == "modified")
      pStats_ = permStats_(permOut,outputData);
  }//end of if (permAnalysis==true)
}//end of else no tokens

/************************
* 5. print results to file
***/
//************
//Main Results
String outputPath = f.getParent() + "\\AS.out";
BufferedWriter bw = new BufferedWriter(new FileWriter(outputPath));
if(outputData[0].isEmpty()){
  bw.newLine();
  bw.write("There is a different number of columns of input data found compared to
  that identified by the user");
} else {
  for(int i=0;i<outputData[0].size();i++){
    bw.write(((Double)outputData[0].get(i)).toString());
    bw.write(",");
    bw.write(((Double)outputData[1].get(i)).toString());
    bw.write(",");
    bw.write(((Double)outputData[2].get(i)).toString());
    bw.newLine();
  }//end of for
}//end of else
bw.close();

//******************
//Permutation Results
if(permAnalysis == true){
  String outputPathPermStats = f.getParent() + "\\AS_permutations.stats";
  BufferedWriter bwPermStats = new BufferedWriter(new FileWriter
  (outputPathPermStats));
  bwPermStats.write("Permutation Analysis Results");
  bwPermStats.newLine();
  bwPermStats.write("====================");
  bwPermStats.write("\n\nPermutation Type: " + permType);
  bwPermStats.write("\nNumber of Permutations: " + noPerms);
  bwPermStats.write("\nOriginal data from file: " + inputPath);
  bwPermStats.write("\nResults of the analysis of the real data can be found in
  file: AS.out");
```

```
    bwPermStats.write("\nRecord of the randomisation of haplotypes can be found in
    file: permutations.log");
    bwPermStats.write("\nRecord of the actual results of each permutation can be
    found in file: Perm_diffs_all.txt");

    if (permType == "standard"){
      bwPermStats.write("\n-----------------------------------------------------------
      ------------------");
      bwPermStats.write("\nColumn 1 shows (for each marker) the proportion of
      permutations that were gave the difference in allele sharing greater than that
      seen in the real results\n");
      for(int i=0;i<pStats.size();i++){
        bwPermStats.newLine();
        bwPermStats.write( ((Double)pStats.get(i)).toString() );
      }
    } else if (permType == "modified"){
      bwPermStats.write("\n-----------------------------------------------------------
      ------------------");
      bwPermStats.write("\n\nData shows (for each marker) the proportion of
      permutations that were gave allele sharing greater than that seen in the real
      results for cases");
      for(int i=0;i<pStats_.size();i++){
        bwPermStats.newLine();
        bwPermStats.write( ((Double)pStats_.get(i)).toString() );
      }
    }
    bwPermStats.close();

    //******
    //also want to print out the difference data for EVERY marker
    //- for sig analysis
    String outputPathPermDiffs = f.getParent() + "\\AS_permutations.all";
    BufferedWriter bwPermDiffsAll = new BufferedWriter(new
    FileWriter(outputPathPermDiffs));
    //if we are using standard perm analysis - print to file, the diffs of
    //each permutation
    if(permType == "standard"){
      for (int i=0;i<(permOut.length);i++){
        for (int j=0;j<((Vector)(permOut[i])).size();j++){
          bwPermDiffsAll.write(  ((Double)permOut[i].get(j) ).toString() );
          bwPermDiffsAll.write(",");
        }
        bwPermDiffsAll.newLine();
      }
    }
    //if we are using modified perm analysis - print to file the actual
    //sharing for each permutation (no diffs cause we do not calc for ctrls)
    else if (permType == "modified"){
      for (int i=0;i<(permOut.length);i++){
        for (int j=0;j<((Vector)(permOut[i])).size();j++){
          bwPermDiffsAll.write(  ((Double)permOut[i].get(j) ).toString() );
          bwPermDiffsAll.write(",");
        }
        bwPermDiffsAll.newLine();
      }
    }
    bwPermDiffsAll.close();
  }//END OF PERM ANALSIS OUTPUT (if requested)
}

/***
*          _calcAS_ method to carry out AS calculation
*          takes the input data and the set of colums that are relevant
*          carries out pairwise comparison for each of them
*          returns a vector of the results.
***/
private Vector calcAS(Vector inputData, int colStart, int colEnd){
  Vector pairCases = new Vector();//create a vector of each pairwise comparison
  //create a loop of nocases factorial(!)
  for (int i=colStart;i<colEnd;i++){
    for (int j=i+1;j<colEnd;j++){
      //whithin this loop, carry out the pairwise comparison, and enter into
      //the first row of pairComp vector
      Vector pairTemp = new Vector();//new vector for each parwise comparison
      if (ASType == 1)//meansuring length in bp distance
        pairTemp = ASBp((Vector)inputData.get(i),(Vector)inputData.get(j));
      else if (ASType == 0)//measuring length as number of alleles
```

```
        pairTemp = ASLength((Vector)inputData.get(i),(Vector)inputData.get(j));
      //add the result of each pairwise comparison to the pairComp vector
      pairCases.add(pairTemp);
    }
  }//end of loop around the no of pairwise comparisons
  return pairCases;
}//end of calcAS

private Vector ASBp(Vector col1,Vector col2){
  Vector pairTemp = new Vector();
  int n=0;
  int m=0;
  for (int k=0;k<col1.size();k++){
    if ( ((Double)(col1.get(k))).doubleValue() == -1.0   ){
      pairTemp.add(new Double(-111111.0));
      m++;//if missing in col1 m++
    } else if ( ((Double)(col2.get(k))).doubleValue() == -1.0   ){
      pairTemp.add(new Double(-111111.0));
      m++;//if missing in col2 m++
    } else if ((col1.get(k)).equals(col2.get(k))){
      pairTemp.add(new Double(-111111.0));
      n++;//if matching, n++
    } else {
      //when there is no match, or no missing data, go back and fill in the distace
      //for the previos string of matches, and also add a zero entry for this
      //non-match
      double distTemp = 0.0;
      if(n>1)
        distTemp = MRI[k-1] - MRI[k-(n+m)];
      for(int l=0;l<(n+m);l++)
          pairTemp.set((k-l-1),new Double(distTemp));
      pairTemp.add(new Double(0.0));
      n=0;
      m=0;
    }
  }//end of for
  if(m>0 || n>0){//if the loop has not ended on a non-match, need to tidy up
    if(n>1){
      double distTemp = MRI[col1.size()-1] - MRI[col1.size()-n-m];
      for(int l=0;l<(n+m);l++)
        pairTemp.set((col1.size()-l-1),new Double(distTemp));
    } else{
      for(int l=0;l<(n+m);l++)
        pairTemp.set((col1.size()-l-1),new Double(0.0));
    }
  }
  return pairTemp;
}

//****
//*         calcAS sub-method - length measure
//****
private Vector ASLength(Vector col1,Vector col2){
  Vector pairTemp = new Vector();
  int n=0;//to count where strings of matches occur
  int m=0;//take into account the number of missing points
  int p=0;//counts the number of matches
  //keep track of the mounting score allocated from abiguous matches
  double pscore=0.0;
  for (int k=0;k<col1.size();k++){
    if (missingType == 0){//ignore missing, compare abmbiguous method
      String st1 = (String)col1.get(k);
      String st2 = (String)col2.get(k);
      if (st1.equals("?") || st2.equals("?")){//data missing in hap1 or hap2
        pairTemp.add(new Double(n+pscore));
        if(n>0)//this is important - it means that if there is a ?, it only gets a
               //score if there is a score immediately prior, even if there is one
               //immediately after - this can be changed
          m++;
      } else if(!isInteger(st1) || !isInteger(st2)){//need to do pairwise if either
                                                    //hap1 or hap2 are ambiguous
                                                    //genotypes
        double match = 0;
        if (!isInteger(st1)){//if the hap1 genotype also ambigous
          String st1s[] = st1.split("_");//get the two options
          if (!isInteger(st2)){//if the hap2 is also ambiguous
            String st2s[] = st2.split("_");//get the hap2 options
```

```
            for (int i=0;i<st1s.length;i++){
              for (int j=0;j<st2s.length;j++){
                //check that thas part of the ambiguity for either hap is not '?'
                if (isInteger(st1s[i]) && isInteger(st2s[j])){
                  if ( (new Double(st1s[i])).equals(new Double(st2s[j])) )
                    match += 1.0;
                }
              }//end of for st2s.length
            }//end of for st1s.lenght
            match /= 4.0;
          } else {//only hap1 is ambiguous
            for (int i=0;i<st1s.length;i++){
              //check that this part of the ambiguity is not '?'
              if (isInteger(st1s[i])){
                if ( (new Double(st1s[i])).equals(new Double(st2)) )
                  match += 1.0;
              }
            }
            match /= 2.0;
          }
        } else {//it must be that only hap2 is ambiguous
          String st2s[] = st2.split("_");//get the hap2 options
          for (int i=0;i<st2s.length;i++){
            //check that this part of the ambiguity is not '?'
            if (isInteger(st2s[i])){
              if ( (new Double(st1)).equals(new Double(st2s[i])) )
                match += 1.0;
            }
          }
          match /= 2.0;
        }
        if (match > 0.0){
          pscore += match;
          pairTemp.add(new Double(n+pscore));
          for(int l=0;l<(n+p+m);l++){
            pairTemp.set((k-l-1),new Double(n+pscore));
          }
          p++;
        } else if (match == 0.0){//there is no match even from the ambiguous data
          pairTemp.add(new Double(0.0));
          n=0; m=0; p=0; pscore=0.0;
        }
      } else if ( (new Double(st1)).equals(new Double(st2))   ){
        pairTemp.add(new Double(n+pscore+1));
        for(int l=0;l<(n+p+m);l++){
          pairTemp.set((k-l-1),new Double(n+pscore+1));
        }
        n++;
      } else {
        pairTemp.add(new Double(0.0));
        n=0; m=0; p=0; pscore=0.0;
      }
    }
    else if (missingType == 1){//ignore missing and ambiguous method
      if ( ((Double)(col1.get(k))).doubleValue() == -1.0   ){
        pairTemp.add(new Double(n));
        if(n>0)
          m++;
      } else if ( ((Double)(col2.get(k))).doubleValue() == -1.0   ){
        pairTemp.add(new Double(n));
        if(n>0)
          m++;
      } else if ((col1.get(k)).equals(col2.get(k))){
        pairTemp.add(new Double(n+1.0));
        for(int l=0;l<(n+m);l++){
          pairTemp.set((k-l-1),new Double(n+1.0));
        }
        n++;
      } else {
        pairTemp.add(new Double(0.0));
        n=0; m=0;
      }
    }
  }//end of loop around the lenght of the pair of columns
  return pairTemp;
}
```

```
  //****
  //*         method to take the average of a group of vectors
  //****
  private Vector average(Vector pairCases){
    Vector av = new Vector();
    for (int i=0;i<((Vector)pairCases.get(0)).size();i++){
      double sum = 0;
      for(int j=0;j<(pairCases.size());j++){
        sum += ((Double)(((Vector)pairCases.get(j)).get(i))).doubleValue();
      }
      double avg = sum/pairCases.size();
      av.add(new Double(avg));
    }
    return av;
  }

  /**
   *    Calculate the difference between content of two vectors
   */
  private Vector difference(Vector cases, Vector ctrls){
    Vector diff = new Vector();
    for (int i=0;i<cases.size();i++){
      diff.add(      new      Double(      ((Double)cases.get(i)).doubleValue()      -
((Double)ctrls.get(i)).doubleValue()) );
    }
    return diff;
  }

  /**
   *      Method to test if a string is an integer
   */
  private final boolean isInteger( String s ){
    try{
      Integer d = new Integer( s );
      return true;
    }catch(NumberFormatException e){
      return false;
    }//end try/catch
  }//end isInteger()

  /**
   *    Get a random no
   */
  private static int getRnd(int nCealing, Random r) {
      int nRet = r.nextInt();
      nRet = Math.abs(nRet);
      nRet = nRet % nCealing;
      return nRet;
  }

  /**
   *
   */
  private Vector permStats_(Vector[] perm, Vector[] out){
    Vector statsSum = new Vector();
    for (int i=0;i<perm[0].size();i++){
      int temp = 0;
      double nCase = 0.0;
      for (int j=0;j<perm.length;j++){
        if( ((Double)perm[j].get(i)).doubleValue() < ((Double)out[0].get(i))
        .doubleValue() )
          nCase++;
      }//end of for perm.length
      nCase /= noPerms;
      statsSum.add(new Double(nCase));
    }
    return statsSum;
  }

  private Vector permStats(Vector[] perm, Vector[] out){
    //store the no of permutations that are less than what is seen in the real results
    Vector vDiffs = new Vector();
    for (int i=0;i<perm[0].size();i++){
      int temp = 0;
      double nDiffs = 0.0;
      for(int j=0;j<perm.length;j++){
          if(((Double)(perm[j].get(i))).doubleValue()< ((Double)(out[2].get(i))).
```

```
          doubleValue() )
            nDiffs++;//CHANGE THIS
          temp = 0;//diff columnn
      }//end of for perm.length (columns)

      nDiffs /= noPerms;
      vDiffs.add(new Double(nDiffs));
    }//end of for perm[0].size (rows)
    return vDiffs;
  }

  //publicly accesible method that returns the progress made in carrying out the
  //permutation analysis
  public int getProg(){
    return progCount;
  }
}
```

A.2 NewASPermAnalysis

```
import java.io.*;
import java.util.StringTokenizer;
import java.util.Vector;

public class NewASPermAnalysis {

  //store the marker position of the end of each block
  Vector endofblock = new Vector();
  //and the size of each block
  Vector blocksize = new Vector();
  //the number of consecutive blocks that make up the best region of sharing
  int testNBlocks;
  //and the average score for these blocks in the real data
  double realAvScore;

  public NewASPermAnalysis(String inputPath, String inputPath2, String inputPath3) {
    //****************
    //1. Get the data
    //inputPath1 is AS.out
    //get the data
    Vector realData = new Vector();
    try {
      BufferedReader br = new BufferedReader(new FileReader(inputPath));
      String line = br.readLine();
      while (line != null){
        StringTokenizer tok = new StringTokenizer(line,",");
        tok.nextToken();//ignore first token
        tok.nextToken();//ignore second token
        realData.add(tok.nextToken());//store the third token
        line = br.readLine();
      }
    } catch (Exception e){
      System.out.println("Error reading from AS.out file: " + e);
    }
    //inputPath2 is AS_permutations.all
    //get this data
    Vector permData = new Vector();
    try {
      BufferedReader br = new BufferedReader(new FileReader(inputPath2));
      String line = br.readLine();
      while (line != null){
        Vector permDataTemp = new Vector();
        StringTokenizer tok = new StringTokenizer(line,",");
        while (tok.hasMoreTokens())
          permDataTemp.add(tok.nextToken());//
        line = br.readLine();
        permData.add(permDataTemp);
      }
    } catch (Exception e){
      System.out.println("Error reading from AS_permutations.all file: " + e);
    }
    //inputPath3 is data.dat
    //get this data
    try {
       BufferedReader br = new BufferedReader(new FileReader(inputPath3));
       String line1 = br.readLine();
       StringTokenizer tok = new StringTokenizer(line1,",");
       while (tok.hasMoreTokens())
         endofblock.add(tok.nextToken());
       String line2 = br.readLine();
       StringTokenizer tok2 = new StringTokenizer(line2,",");
       while (tok2.hasMoreTokens())
         blocksize.add(tok2.nextToken());
       String line3 = br.readLine();
       testNBlocks = new Integer(line3).intValue();
       String line4 = br.readLine();
       realAvScore = new Double(line4).doubleValue();
    } catch (Exception e){
      System.out.println("Error reading from data.dat file: " + e);
    }
    //end of 1.
```

```
//***********
//2. now we want to compare
//first, get the average score for each block, for the real data
Vector realBlockData = getBlockAvs(realData);
//secondly, store the average scores or each block, for each permutation
Vector permBlockData = new Vector();
//for each permutation
for (int i=0;i<permData.size();i++){
  permBlockData.add(getBlockAvs((Vector)permData.get(i)));
}
//2.b. we also want to then go on and take the average of each seq of N
//blocks for each permutation
Vector permTestNBlocksData = new Vector();
for (int i=0;i<permBlockData.size();i++){
  permTestNBlocksData.add(new Double(getBestNBlockScore(
  (Vector)permBlockData.get(i))));
}
double bestScore = 0.0;
for (int i=0;i<permTestNBlocksData.size();i++){
  if(  ((Double)permTestNBlocksData.get(i)).doubleValue() >= bestScore){
    bestScore = ((Double)permTestNBlocksData.get(i)).doubleValue();
  }
}

//******************
//3. compare the real data to each of the permutations
Vector realBlockPValues = new Vector();
for (int k=0;k<((Vector)permBlockData.get(0)).size();k++){//for each block
  double score = 0.0;
  for (int i=0;i<permBlockData.size();i++){//for each permutaiton
    if ( ((Double)((Vector)permBlockData.get(i)).get(k)).doubleValue() > ((Double)
    (realBlockData).get(k)).doubleValue()  ){
          //add 1.0 to the score if the permutations score is > the realData score
          score += 1.0;
    }
  }//end of for each permutation
  //now calc the proportion of the permutationsdata > real data
  double propn = score/permBlockData.size();
  //add the propn (or P value) to the p-values vector
  realBlockPValues.add(new Double(propn));
}//end of each marker
//3.b. we also want to compare the best score for N consecutive blocks in
//each of the permutations against the real result
int noMoreSigPerms = 0;
//for each perm
for (int i=0;i<permTestNBlocksData.size();i++){
  //now compare
  if(  ((Double)permTestNBlocksData.get(i)).doubleValue() >= realAvScore){
    //and score as the number of significant permutations
    noMoreSigPerms++;
  }
}

//4. compare the results for each permutation to all the others
Vector permBlockPValues = new Vector();
for (int k=0;k<((Vector)permBlockData.get(0)).size();k++){//for each block
  Vector permBlockPValuesTemp = new Vector();
  for (int i=0;i<permBlockData.size();i++){//for each datai (permutaiton)
    double score = 0.0;
    //go though all the rest j (other permutations)
    for (int j=0;j<permBlockData.size();j++){
      if (i!=j){//apart from itself
        //if datai>dataj
        Vector v = (Vector)permBlockData.get(i);
        if ( ((Double)((Vector)permBlockData.get(j)).get(k)).doubleValue() >
        ((Double)((Vector)permBlockData.get(i)).get(k)).doubleValue()  ){
          score += 1.0;//add 1.0 to the score
        }
      }
    }//end of all other perms
    //now calc the proportion of datai > dataj
    double propn = score/permBlockData.size();
    //add the propn (or P value) to pOut vector
    permBlockPValuesTemp.add(new Double(propn));
  }//end of for each permutation
  permBlockPValues.add(permBlockPValuesTemp);
```

```
  }//end of each marker

  //5. write these results to file
  //first the reaData vs permData
  String outputPath = (new File(inputPath3)).getParent().
  concat("//newASRealPermutationsPValues.out");
  try{
    BufferedWriter bw = new BufferedWriter(new FileWriter(outputPath));
    bw.write("");
    for (int i=0;i<realBlockPValues.size();i++){
      bw.write("" + realBlockPValues.get(i));
      bw.write("\n");
    }
    bw.close();
  } catch (Exception e){
    System.out.println("Error writing to file: " + e);
  }
  //then the permData vs all other perms
  String outputPath2 = (new File(inputPath3)).getParent().
  concat("//newASPermPermutationsPValues.out");
  try{
    BufferedWriter bw = new BufferedWriter(new FileWriter(outputPath2));
    bw.write("");
    for (int i=0;i<permBlockPValues.size();i++){
      bw.write("" + ((Vector)permBlockPValues.get(i)).get(0));
      for (int j=1;j<((Vector)permBlockPValues.get(i)).size();j++){
        bw.write("," + ((Vector)permBlockPValues.get(i)).get(j));
      }
      bw.write("\n");
    }
    bw.close();
  } catch (Exception e){
    System.out.println("Error writing to file: " + e);
  }
  //also add a new file that outputs the scores for each permutation for each
  //block
  String outputPath3 = (new File(inputPath3)).getParent().
  concat("//newASPermPermutationsScores.out");
  try{
    BufferedWriter bw = new BufferedWriter(new FileWriter(outputPath3));
    bw.write("");
    for (int i=0;i<permBlockData.size();i++){
      bw.write("" + ((Vector)permBlockData.get(i)).get(0));
      for (int j=1;j<((Vector)permBlockData.get(i)).size();j++){
        bw.write("," + ((Vector)permBlockData.get(i)).get(j));
      }
      bw.write("\n");
    }
    bw.close();
  } catch (Exception e){
    System.out.println("Error writing to file: " + e);
  }
  //finally, write the results where we compared the av scores for n consec
  //blocks
  String outputPath4 = (new File(inputPath3)).getParent().
  concat("//new2ndPermAnalysis.out");
  try{
    BufferedWriter bw = new BufferedWriter(new FileWriter(outputPath4));
    bw.write("There were " + noMoreSigPerms);
    bw.write(" permutations where there was a sequence of " + testNBlocks);
    bw.write(" blocks that showed an average allele sharing score >= that found in
    the real data");
    bw.close();
  } catch (Exception e){
    System.out.println("Error writing to file: " + e);
  }
}

/**
 * Get block averages
 *
 * this is a method to go through a vector of data where each cell is a marker
 * and calculate a score for each block
 */
private Vector getBlockAvs(Vector markerData){
  //keep track of the block we are at
  int blockCount = 0;
```

```
  //sum of datapts in a block
  double dataptSum = 0.0;
  //store the av score for each LD block
  Vector blockScores = new Vector();
  //for each marker
  for (int i=0;i<markerData.size();i++){
    double datapt = (new Double((String)markerData.get(i))).doubleValue();
    //add up the scores for all blocks, and average
    dataptSum += datapt;
    //if we are at the end of a block
    if(i == (new Integer((String)endofblock.get(blockCount))).intValue() ) {
      //get the average datapt for the block
      double dataptAv = dataptSum/( (new
      Integer((String)blocksize.get(blockCount))).intValue() );
      blockScores.add(new Double(dataptAv));
      blockCount++;
      dataptSum = 0.0;
    }
  }
  return blockScores;
}

/**
 * Get best score for N consecutive blocks
 *
 * this is a method to go through a vector of data where each cell is a block
 * and calculate the best average score for a sequence of N consequtive blocks
 */
private double getBestNBlockScore(Vector blockData){
  //go through each of the list of block averages
  //for te first n-1 add to a vector
  //then for n go on and take the average again
  //then for each additional block remove the first and add the new and take
  //another average
  Vector consecBlocks = new Vector();
  Vector avBlockScores = new Vector();
  double bestNBlockScore = 0.0;
  for (int i=0;i<blockData.size();i++){
    //if we have been through less than the required number of blocks so far
    //add the latest block to the list
    if(consecBlocks.size() < (testNBlocks-1)){
      consecBlocks.add(blockData.get(i));
    }//else if we have just 1 less than required
    else if (consecBlocks.size() == (testNBlocks-1)){
      //add the newest block
      consecBlocks.add(blockData.get(i));
      //and calc the average score
      double sum = 0.0;
      for (int k=0;k<testNBlocks;k++){
        sum += ((Double)consecBlocks.get(k)).doubleValue();
      }
      bestNBlockScore = sum/testNBlocks;
    }//else we then add new block to the end and remove the oldst block from
    //the start
    else {
      //remove the oldest blcock
      consecBlocks.remove(0);
      //add the newest block
      consecBlocks.add(blockData.get(i));
      //and calc the average score
      double sum = 0.0;
      for (int k=0;k<testNBlocks;k++){
        sum += ((Double)consecBlocks.get(k)).doubleValue();
      }
      if ( (sum/testNBlocks) > bestNBlockScore)
        bestNBlockScore = sum/testNBlocks;
    }//end of final else
  }
  return bestNBlockScore;
}
```

A.8 PedSimMain

```
import java.io.*;
import java.lang.Math;
import java.lang.Thread;
import java.util.StringTokenizer;
import java.util.Random;
import java.util.Vector;
import java.util.StringTokenizer;
import java.util.Hashtable;
import java.util.Enumeration;

public class PedSimMain {


  /******
   * Variables
   */
  //number of simlations
  private int noSims = 200;
  //number of generations
  private int noGens = 50;
  //no of end families we want to generate
  private int noFams = 3;
  //store the recombination rate (chance per base that there is a recomb event)
  private double recombRate = 1.0E-8;
  //dist between markers (in bps)
  private double markerDist = (16778.0*150.0);
  //number of markers
  private int noMarkers = 149;

  //number of permutations to carry out the AS analysis over
  private int noASPerms = 1000;

  /*****
   *
   */
  //store the location of the mutation on the disease chr
  private int mutMarker;
  //folder in which to work
  private String infolder = "//net//uisdein//export//usr0//s9636861//code//sim2//";
  //create a miclcalculate object to run the AS
  private ASCalculate mc;
  //create a newASpermanalysis object to run the additional perm analysis
  private NewASPermAnalysis nmpa;

  public PedSimMain() {

    //1. getData - starting data
    int[][] population = importHaps();
    //and then generate a log file with the sim details
    try{
      BufferedWriter bwLog = new BufferedWriter(new FileWriter(infolder +
      "AllStatsSummary.txt"));
      bwLog.write("Simulation study");
      bwLog.newLine();
      bwLog.write("Number of simulations:" + noSims);
      bwLog.newLine();
      bwLog.write("Number of families generated:" + noFams);
      bwLog.newLine();
      bwLog.write("Number of AS permutations:" + noASPerms);
      bwLog.newLine();
      bwLog.write("Recombination rate:" + recombRate);
      bwLog.newLine();
      bwLog.write("Dist between markers (evenly spaced):" + markerDist);
      bwLog.newLine();
      bwLog.write("Number of markers:" + (population[0].length) );
      bwLog.close();
    } catch (Exception e){
    }
    //carry out a number of sims of this population data
    //place each of these sims in a separate folder
    //in each of these folders, the resuts of the sim are analysed using AS
    //and then the additional perm analysis
```

```
    //this means that for each sim, we generatate a p value for the region about the
    mutation
    //and we also look at any other significant regions that appear

    //2. for each sim
    for (int i=0;i<noSims;i++){
    String outfolder = "//net//uisdein//export//usr0//s9636861//code//sim2//sim" + i +
    "//";
    File f = new File(outfolder);
    //if the folder doesn't already exist, make it
    if(!f.exists())
      f.mkdir();
    //list files in the folder
    File[] files = f.listFiles( new FileFilter(){
      public boolean accept(File f){
          return true;
      }
    } );
    //go through and delete all these files
    for (int j=0;j<files.length;j++){
      if( files[j].isDirectory() ){
        File [] files2 = files[j].listFiles();
        for (int k=0;k<files2.length;k++){
          files2[k].delete();
        }
      }
      files[j].delete();
    }

    //3. the sim
    //identify one of these to be the disease hap
    double x = population.length;
    double rand = Math.random();
    int diseaseHapIndex = (int)(rand*x);
    int[] diseaseHap = population[diseaseHapIndex];
    //specify the markers closest to the disease mutation
    //at present assume that the central marker is the one **********************
    mutMarker = diseaseHap.length/2;
    //send data to similate class
    //keep trying until a successful simulation is carried out
    //this sim creates (amongst others) a ASIn.csv file
    boolean status = false;
    int j=0;
    int noASCtrls = 0;
    while (!status){
      Simulator2 sim = new Simulator2(diseaseHap, population, noGens, noFams,
      markerDist, mutMarker, recombRate, outfolder);
      status = sim.getStatus();
      noASCtrls = sim.getASNoCtrls();
    }

    //4. use inputs to calc AS
    String ASInPath = outfolder + "ASIn.csv";
    try{
      mc = new ASCalculate(ASInPath,noFams,noASCtrls,0,1,true,"standard"
      ,noASPerms);
      mc.jbInit();
    } catch (Exception ioe){
      System.out.println("IOException e: " + ioe);
    }

    //5.the additional perm analysis
    //use the AS.out file
    String ASOutPath = outfolder + "AS.out";
    //and the AS_permutations.all file
    String ASPermsAllPath = outfolder + "AS_permutations.all";
    //and the AS_permutations.stats file
    String ASStatsPath = outfolder + "AS_permutations.stats";
    //and the MyHapMap.out file
    String blockDefPath = infolder + "dat//MyHapMap.out";
    //and the data.dat file
    String datPath = infolder + "dat//data.dat";
    //need to go through the ASIn.csv file and find all the shared regions
    FindSigRegions                    fsr                    =                    new
FindSigRegions(ASInPath,ASOutPath,ASStatsPath,blockDefPath,datPath,mutMarker);
    //the fsr method generates muliple data.dat files allowing the add perm analysis
    //to be carried out on each of them.
```

```
//run the add perm analysis
int n=0;
String datFile = outfolder  + "sr" + n + "//data.dat";
while( (new File(datFile).exists()) ){
  nmpa = new NewASPermAnalysis(ASOutPath,ASPermsAllPath,datFile);
  n++;
  datFile = outfolder  + "sr" + n + "//data.dat";
}

/*********************************************
 **  6.log the sig regions
 */
Vector sigValues = new Vector();
Vector sigRegBlocks = new Vector();
int mutIndex_temp = 1000000;
n=0;
//count the non-sig regions
int nonCount = 0;
datFile = outfolder  + "sr" + n + "//new2ndPermAnalysis.out";
while( (new File(datFile).exists()) ){
  try {
    BufferedReader br = new BufferedReader(new FileReader(datFile));
    String st = br.readLine();
    StringTokenizer tok = new StringTokenizer(st,"\t");
    double sigValues_temp = (new Double((String)tok.nextToken())).doubleValue();
    if (sigValues_temp <= (0.05*noASPerms)){
      sigValues.add(new Double(sigValues_temp));
          sigRegBlocks.add(tok.nextToken());
    } else {
      nonCount++;
    }
    br.close();
  } catch (Exception e){
    System.out.println("Error reading new2ndPermAnalysis.out: " + e);
  }
  if (  (new File(outfolder + "sr" + n + "//MUTATION")).exists()  ){
    mutIndex_temp = (n-nonCount);
  }
  n++;
  datFile = outfolder  + "sr" + n + "//new2ndPermAnalysis.out";
}
//delete content of sr folder
n=0;
File srFolder = new File(outfolder + "sr" + n);
while ( srFolder.exists() ){
  File[] contentSRFolder = srFolder.listFiles(new FileFilter(){
    public boolean accept(File f){
      return true;
  }});
  for (int k=0;k<contentSRFolder.length;k++){
    contentSRFolder[k].delete();
  }
  n++;
  srFolder = new File(outfolder + "sr" + n);
}
try {
  BufferedWriter bw = new BufferedWriter(new FileWriter((outfolder +
  "StatsSummary.txt")));
  bw.write("SR,P,IsMutReg,NoBlocks");
  bw.newLine();
  for (int k=0;k<sigValues.size();k++){
    bw.write(k + "," + sigValues.get(k) + ",");
    if (mutIndex_temp == k){
      bw.write("1,");
    } else {
  bw.write("0,");
}
    bw.write("" + sigRegBlocks.get(k) );
    bw.newLine();
  }
  bw.close();
} catch (Exception e){
  System.out.println("Error writing to StatsSummary.txt: " + e);
}
sigValues.removeAllElements();
sigRegBlocks.removeAllElements();
}//END OF FOR EACH SIM
```

```
  //call SummarStats
  SummarStats ss = new SummarStats();
  for (int i=0;i<noSims;i++){
    String simFolder = infolder + "sim" + i;
    (new File(simFolder)).delete();
    }
  }

  //import the haplotypes
  //
  //each line is a haplotype
  //
  private int[][] importHaps(){
    int[][] haps = new int[916][noMarkers];
    try {
      BufferedReader br = new BufferedReader(new FileReader(infolder +
      "hapsin//regb_haps.in"));
      String line = br.readLine();
      int i=0;
      while (line!=null){
        StringTokenizer tok = new StringTokenizer(line,",");
        int[] hapTemp = new int[noMarkers];
        int j=0;
        while (tok.hasMoreTokens()){
          hapTemp[j++] = (new Integer(tok.nextToken())).intValue();
        }
        haps[i++] = hapTemp;
        line = br.readLine();
      }
      br.close();
    } catch (Exception e){
      System.out.println("Error importing haps: " + e);
    }
    return haps;
  }

  public static void main(String[] args) {
    PedSimMain psm = new PedSimMain();
  }
}

class Simulator2 {

  /**
   * global variables
   */
  private int noFams;//store the final no of fams we want to generate
  private boolean status;//use to infom main if simulation successful
  private double distBetweenMarkers;//dist between each marker
  private int mutMarker;//store the location of the mutation on the disease chr
  private double recombRate;//store the rate of recombination
  private String infolder;//location of job
  private int noASCtrls;//store the number of control haps in the ASIn file
  private Hashtable generationsHT;//
  private Hashtable peopleHT_;//
  private Hashtable prevGenHT;//
  private Hashtable peopleHT;//

  public Simulator2(int[] diseaseHap, int[][] controlHaps, int noGenerations, int
noFams, double distBetweenMarkers, int mutMarker, double recombRate, String infolder)
{
    this.noFams = noFams;
    this.distBetweenMarkers = distBetweenMarkers;
    this.mutMarker = mutMarker;
    this.recombRate = recombRate;
    this.infolder = infolder;

    /*****
     * Initialise
     */
    //set status = false
    setStatus(false);
    //select a disease chromosome
    int[] hapDisease = diseaseHap;
    //to start, call the original affected individual the first offspring
    //they will carry the the disease haplotype and 1 control chromosome
    peopleHT_ = new Hashtable();
```

```
//for the first affected individual
//store the two haplotypes (chromsomes) and the affection status.
Vector affectedPerson = new Vector();
affectedPerson.add("N,N");//parentID (N is no parent)
affectedPerson.add("D");//disease status
affectedPerson.add(hapDisease);//haplotpye1
affectedPerson.add(controlHaps[(int)Math.floor(Math.random()*
(controlHaps.length))]);//random haplotype2
peopleHT_.put("0",affectedPerson);
//for the first married in
Vector minPerson = makeMarriedIn(controlHaps[(int)Math.floor(Math.random()*
(controlHaps.length))],controlHaps[(int)Math.floor(Math.random()*
(controlHaps.length))]);
peopleHT_.put("1",minPerson);
//create hashtable to store results
generationsHT = new Hashtable();
//add the first generation
generationsHT.put("0",peopleHT_);
//keep count of all the individuals
int idCount = 2;

/*********************************
 *
 * Actual sim
 *
 **/
//1. for each generation
for (int i=1;i<noGenerations;i++){
  //store the data for each generation in a new hashtable
  peopleHT = new Hashtable();
  //get the info from the previous generation
  prevGenHT = (Hashtable)generationsHT.get((new Integer(i-1)).toString());

  //2. for each parent
  Vector sortedKeys = new Vector();
  sortedKeys = sort(prevGenHT.keys());
  for (int j=0;j<sortedKeys.size();j++){
    /**
     * Here we model the recombination
     */
    //initialise child1
    Vector child0 = new Vector();
    Vector child1 = new Vector();
    //select 50/50 whether to start from parent0 chr 0 or 1
    String key = ((Integer)sortedKeys.get(j)).toString();
    Vector v = new Vector();
    v = (Vector)prevGenHT.get(key);
    //store affection status from 1st parent
    String parentDS = (String)v.get(1);
    //child0
    Vector c0c0temp = new Vector();
    c0c0temp = getChildChr((int[])v.get(2),(int[])v.get(3),(String)v.get(1),key);
    int[] child0Chr0 = (int[])c0c0temp.get(0);
    String child0DS = (String)c0c0temp.get(1);
    //child1
    Vector c1c0temp = new Vector();
    c1c0temp = getChildChr((int[])v.get(2),(int[])v.get(3),(String)v.get(1),key);
    int[] child1Chr0 = (int[])c1c0temp.get(0);
    String child1DS = (String)c1c0temp.get(1);
    //
    j++;
    String key2 = ((Integer)sortedKeys.get(j)).toString();
    Vector v2 = (Vector)prevGenHT.get(key2);
    //child0
    Vector c0c1temp = new Vector();
    int[] t = (int[])v2.get(2);
    int[] t2 = (int[])v2.get(3);
    String t3 = (String)v2.get(1);
    c0c1temp = getChildChr((int[])v2.get(2),(int[])v2.get(3),
    (String)v2.get(1),key);
    int[] child0Chr1 = (int[])c0c1temp.get(0);
    //child1
    Vector c1c1temp = new Vector();
    c1c1temp = getChildChr((int[])v2.get(2),(int[])v2.get(3),
    (String)v2.get(1),key);
    int[] child1Chr1 = (int[])c1c1temp.get(0);
```

```
      //4.
      child0.add(key.concat(",").concat(key2));//parentID
      child1.add(key.concat(",").concat(key2));//parentID
      child0.add(child0DS);//disease status
      child1.add(child1DS);//disease status
      child0.add(child0Chr0);//haplotpye1
      child1.add(child1Chr0);//haplotpye1
      child0.add(child0Chr1);//haplotype2
      child1.add(child1Chr1);//haplotype2

      //if the parent is affected
      if(parentDS.equals("D")){
        //put child1 into the gen
        peopleHT.put(new Integer(idCount++).toString(),child0);

        //if this is not the last generation, make a married in and add
        if(!(i==(noGenerations-1))){
          Vector min = new Vector();
          min = makeMarriedIn(controlHaps[(int)Math.floor(Math.random()*
          (controlHaps.length))],controlHaps[(int)Math.floor(Math.random()*
          (controlHaps.length))]);
          peopleHT.put(new Integer(idCount++).toString(),min);
        }
        //put child2 into the gen
        peopleHT.put(new Integer(idCount++).toString(),child1);
        if(!(i==(noGenerations-1))){
          Vector min = new Vector();
          min = makeMarriedIn(controlHaps[(int)Math.floor(Math.random()*
          (controlHaps.length))],controlHaps[(int)Math.floor(Math.random()*
          (controlHaps.length))]);
          peopleHT.put(new Integer(idCount++).toString(),min);
        }
      }
    }//end of for each person in prev generation (for sortedkeys)
    //now, put this generations info into gneerationsHT
    if (peopleHT.size()>0){
      generationsHT.put((new Integer(i)).toString(),peopleHT);
    }
    else
      break;
  }//end of for no generation

  /*****************************************************************************
   * Build the families using last 4 generations
   */
  //we must have simulated at least 5 generations to be able to do this
  if(generationsHT.size()==noGenerations){

    /**
     * Split the last four generations into families
     */
    //store a vector of each family
    Vector families = new Vector();
    //Initialise
    Enumeration gkeys = generationsHT.keys();//ennumeration of the key for each
    generation
    Vector gkeysSorted = sort(gkeys);//sorted list of these keys
    //for oldest 0 generaion
    //get the hash table
    Hashtable ht = (Hashtable)generationsHT.get(((Integer)gkeysSorted.get
    (gkeysSorted.size()-4)).toString());
    Enumeration keys = ht.keys();//enumeration of the keys within generation 0
    Vector keysSorted = sort(keys);//these keys sorted
    //for each individual in generation 0
    for (int i=0;i<keysSorted.size();i++){
      //create a family if the person is affected
     if(   ((String)((Vector)ht.get(((Integer)keysSorted.get(i)).toString()))).
     get(1)).equals("D")  ){
        Vector family_temp = new Vector();
        Vector individual_temp = (Vector)((Vector)ht.get(((Integer)keysSorted.
        get(i)).toString())).clone();
        //add the key (individual id) to the start of the vector)
        individual_temp.add(0,(Integer)keysSorted.get(i));
        //now add them to the family temp
        family_temp.add(individual_temp);
        i++;//get that persons spouse
        individual_temp = (Vector)((Vector)ht.get(((Integer)keysSorted.
```

```
    get(i)).toString())).clone();
    //add the key (individual id) to the start of the vector)
    individual_temp.add(0,(Integer)keysSorted.get(i));
    family_temp.add(individual_temp);
    families.add(family_temp);
  } else
    i++;
}
//for each of the subesequent 3 generations
for (int i=(gkeysSorted.size()-3);i<gkeysSorted.size();i++){
  //for generation i
  ht = (Hashtable)generationsHT.get(((Integer)gkeysSorted.get(i)).toString());
  keys = ht.keys();//enumeration of the keys within generation i
  keysSorted = sort(keys);//these keys sorted
  //for each individual j
  for (int j=0;j<keysSorted.size();j++){
    //get j's parents
    String parents = (String)((Vector)ht.get(((Integer)keysSorted.
    get(j)).toString())).get(0);
    String[] parent = parents.split(",");
    //for each family k
    for (int k=0;k<families.size();k++){
      Vector family = (Vector)families.get(k);
      //for each member in that family
      for (int l=0;l<family.size();l++){
        //if j has parents in family k,
        if (parent[0].equals(     ((Integer)((Vector)family.get(l)).
        get(0)).toString()        ) ){
          //get individual j's details
          Integer in = (Integer)keysSorted.get(j);
          Vector individualj = (Vector)((Vector)ht.get(((Integer)keysSorted.
          get(j)).toString())).clone();
          //add the key (individual id) to the start of the vector)
          individualj.add(0,in);
          //add j to that family
          ((Vector)families.get(k)).add(individualj);
          //also add j++ to that family
          Vector individualjplus1 = (Vector)((Vector)ht.get(((Integer)
          keysSorted.get(++j)).toString())).clone();
          individualjplus1.add(0,(Integer)keysSorted.get(j));
          ((Vector)families.get(k)).add(individualjplus1);
          //and break
          break;
        }
      }
    }
  }
}

/**
 * filter families
 * if a family has 6 or less memebers (i.e. only 2 generations large)
 * then is is deleted
 * NB, this is based on the model of two kids per fam
 */
for (int i=0;i<families.size();i++){
  if(((Vector)families.get(i)).size() < 7){
    families.remove(i);
    i--;
  }
}
//only continue if there are at least 3 families
if(families.size()>=3){
  //set status = true
  //this means that we have cheived a result and have generated three
  //families on which to test the allele sharing methods
  setStatus(true);

  /****
   *   now, want to calculate the relatedness of the families
   *
   */
  //create a vector to hold all the comparisons between families
  Vector famDistances = new Vector();
  //for the 1st member of all the family (apart from the last)
  for(int i=0;i<families.size()-1;i++){
    Vector famiid0 = (Vector)((Vector)families.get(i)).get(0);
```

```
  //create a vector to hold the distances between fami and famj
  Vector ijDistances = new Vector();
  //for each other family (exluding comparisons already done)
  //compare parents to those from each of the other families
  //keep going back a generation till you find a common ancestor and store the
  //number of generation difference
  for(int j=i+1;j<families.size();j++){
    //test this person from famiid0 agin famjid0
    Vector famjid0 = (Vector)((Vector)families.get(j)).get(0);
    //initialise parents
    String parents0 = famiid0.get(1).toString();
    String parents1 = famjid0.get(1).toString();
    //store the level of separation
    int sep = 1;
    //go up through all the generations until we find a common ansector
    boolean comAn = false;//is there a com(mon)An(cestor)
    while (!comAn){
      //if they have common parents, then store the no of gen separate
      if( (parents0).equals(parents1)){
        comAn = true;
      }
      //else
      else{
        //go back a further generation
        sep++;
        //and get the parents from the previous generation
        //generationsHT contains info on all the generations
        int noGens = generationsHT.size();
        //we are starting from the last generation
        //so we now want to get the partents of those individuals named as
        //parents 0 and 1
        //so we look at generation (noGens-sep)
        Hashtable generationN = (Hashtable)generationsHT.get((new Integer(
       (noGens-sep-3))).toString());
        //and find the individuals that are named as parents0 and 1 from
        //need to split up the parents0 and 1
        String[] parent00 = parents0.split(",");
        Vector ind0 = (Vector)generationN.get(parent00[0]);
        String[] parent10 = parents1.split(",");
        Vector ind1 = (Vector)generationN.get(parent10[0]);
        //now, get ind0 and 1s parents
        parents0 = (String)ind0.get(0);
        parents1 = (String)ind1.get(0);
        //and loop round again with these new parents
      }
    }
    //store the comparisons
    ijDistances.add(new Integer(sep));
  }
  famDistances.add(ijDistances);
}

/**
 * FIND THE MOST DISTANT FAMILIES
 *
 * select the three most distant families
 * get dist between 1st fam and all others
 * for each of these, calc d(1i)+d(in) where n is last fam
 * calc max of these sums
 * take the three fams that form the max
 */
int max = 0;
//this is the indexs of the matrix that contains the best thrid family
int optIndex = 1;
for(int i=0;i<famDistances.size()-1;i++){
  //if famDistances.get(0).get(i) + famDistances.get(i)(n) > max
  //then max = ...
  //dist between 0 and i
  int dist0i = ((Integer)((Vector)famDistances.get(0)).get(i)).intValue();
  //dist between i and n
  int distin = ((Integer)((Vector)famDistances.get(i+1)).
  lastElement()).intValue();
  if(  (dist0i + distin)  > max){
    max = dist0i + distin;
    optIndex = i+1;
  }
}
```

```
if (((Integer)((Vector)famDistances.get(0)).get(optIndex-1)).intValue() <40 ||
    ((Integer)((Vector)famDistances.get(0)).lastElement()).intValue() <40 ||
    ((Integer)((Vector)famDistances.get(optIndex)).lastElement()).intValue()
    <40  )
  setStatus(false);

/**
 * Write results to file
 */
//write the matrix of distances to file
try {
  BufferedWriter bwDistMat = new BufferedWriter(new FileWriter(infolder +
  "dist.matrix"));
  bwDistMat.write("So, after " + generationsHT.size() + " generations, we have
  " + families.size() + " families.\n" );
  bwDistMat.write("Of which the three most distant families are family 0, " +
  optIndex + " and " + famDistances.size() + ".\n" );
  //write explicitly the distances between the three most distant families
  bwDistMat.write("Distance between fam 0 and " + (optIndex) + " is: " +
 ((Vector)famDistances.get(0)).get(optIndex-1)  + "\n");
  bwDistMat.write("Distance between fam 0 and n is: " +
 ((Vector)famDistances.get(0)).lastElement() + "\n");
  bwDistMat.write("Distance between fam " + (optIndex) + " and n is: " +
 ((Vector)famDistances.get(optIndex)).lastElement());
  bwDistMat.newLine();
  bwDistMat.newLine();
  //write header
  bwDistMat.write("\t");
  for (int i=-1;i<((Vector)famDistances.get(0)).size();i++)
    bwDistMat.write((i+1) + "\t");
  bwDistMat.newLine();
  bwDistMat.newLine();
  //write matrix
  for(int i=0;i<famDistances.size();i++){
    bwDistMat.write("" + i + "\t");
    for(int j=-1;j<i;j++)
      bwDistMat.write("-\t");
    for (int j=0;j<((Vector)famDistances.get(i)).size();j++){
      bwDistMat.write(((Vector)famDistances.get(i)).get(j) + "\t");
    }
    bwDistMat.newLine();
  }
  //
  bwDistMat.write("" + famDistances.size() + "\t");
  for (int i=-1;i<famDistances.size();i++){
    bwDistMat.write("-\t");
  }
  bwDistMat.close();
} catch (Exception e){
  System.out.println("Error writing distance matrix: " + e);
}
//write results - family 1
try {
  BufferedWriter bwFam1 = new BufferedWriter(new FileWriter(infolder +
  "fam1.out"));
  //
  Vector fam1 = (Vector)families.get(0);
  //for each member of the family
  for (int i=0;i<fam1.size();i++){
    Vector inds = (Vector)fam1.get(i);
    //for each characteristic in indi
    bwFam1.write("" + inds.get(0));
    bwFam1.write("," + inds.get(1));
    bwFam1.write("," + inds.get(2));
    int[] hap0 = (int[])inds.get(3);
    for (int j=0;j<hap0.length;j++)
      bwFam1.write("," + hap0[j]);
    bwFam1.newLine();
    bwFam1.write("" + inds.get(0));
    bwFam1.write("," + inds.get(1));
    bwFam1.write("," + inds.get(2));
    int[] hap1 = (int[])inds.get(4);
    for (int j=0;j<hap1.length;j++)
      bwFam1.write("," + hap1[j]);
    bwFam1.newLine();
  }
  bwFam1.close();
```

```
} catch (Exception e){
  System.out.println("Error writing fam1: " + e);
}


//write - most distant third family
try {
  BufferedWriter bwFamI = new BufferedWriter(new FileWriter(infolder +
  "fami.out"));
  Vector fami = (Vector)families.get((optIndex));
  //for each member of the family
  for (int i=0;i<fami.size();i++){
    Vector inds = (Vector)fami.get(i);
    //for each characteristic in indi
    bwFamI.write("" + inds.get(0));
    bwFamI.write("," + inds.get(1));
    bwFamI.write("," + inds.get(2));
    int[] hap0 = (int[])inds.get(3);
    for (int j=0;j<hap0.length;j++)
      bwFamI.write("," + hap0[j]);
    bwFamI.newLine();
    bwFamI.write("" + inds.get(0));
    bwFamI.write("," + inds.get(1));
    bwFamI.write("," + inds.get(2));
    int[] hap1 = (int[])inds.get(4);
    for (int j=0;j<hap1.length;j++)
      bwFamI.write("," + hap1[j]);
    bwFamI.newLine();
  }
  bwFamI.close();
} catch (Exception e){
  System.out.println("Error writing fam1: " + e);
}
//write - familyn (last fam)
try {
  BufferedWriter bwFamN = new BufferedWriter(new FileWriter(infolder +
  "famn.out"));
  Vector famn = (Vector)families.lastElement();
  //for each member of the family
  for (int i=0;i<famn.size();i++){
    Vector inds = (Vector)famn.get(i);
    //for each characteristic in indi
    bwFamN.write("" + inds.get(0));
    bwFamN.write("," + inds.get(1));
    bwFamN.write("," + inds.get(2));
    int[] hap0 = (int[])inds.get(3);
    for (int j=0;j<hap0.length;j++)
      bwFamN.write("," + hap0[j]);
    bwFamN.newLine();
    bwFamN.write("" + inds.get(0));
    bwFamN.write("," + inds.get(1));
    bwFamN.write("," + inds.get(2));
    int[] hap1 = (int[])inds.get(4);
    for (int j=0;j<hap1.length;j++)
      bwFamN.write("," + hap1[j]);
    bwFamN.newLine();
  }
  bwFamN.close();
} catch (Exception e){
  System.out.println("Error writing famN: " + e);
}

//***********************
// contruct a ASIn file
//
Vector ASIn = new Vector();
//add fam1hap0
int[] fam1hap0 = (int[])((Vector)((Vector)families.get(0)).get(0)).get(3);
ASIn.add(fam1hap0);
//add famihap0
ASIn.add((int[])((Vector)((Vector)families.get(optIndex)).get(0)).get(3));
//add famNhap0
ASIn.add((int[])((Vector)((Vector)families.lastElement()).get(0)).get(3));
//add fam1hap1
ASIn.add((int[])((Vector)((Vector)families.get(0)).get(0)).get(4));
//add fam1 married in haps
for (int i=1;i<((Vector)families.get(0)).size();i++){
```

```
        if(((String)((Vector)((Vector)families.get(0)).get(i)).get(1)).
        equals("N,N")){
          ASIn.add((int[])((Vector)((Vector)families.get(0)).get(i)).get(3));
          ASIn.add((int[])((Vector)((Vector)families.get(0)).get(i)).get(4));
        }
      }
      //add famihap1
      ASIn.add((int[])((Vector)((Vector)families.get(optIndex)).get(0)).get(4));
      //add fami married in haps
      for (int i=1;i<((Vector)families.get(optIndex)).size();i++){
        if(  ((String)((Vector)((Vector)families.get(optIndex)).get(i)).get(1)).
        equals("N,N")){
          ASIn.add((int[])((Vector)((Vector)families.get(optIndex)).get(i)).
          get(3));
          ASIn.add((int[])((Vector)((Vector)families.get(optIndex)).get(i)).
          get(4));
        }
      }
      //add famNhap1
      ASIn.add((int[])((Vector)((Vector)families.lastElement()).get(0)).get(4));
      //add famN married in haps
      for (int i=1;i<((Vector)families.lastElement()).size();i++){
        if(  ((String)((Vector)((Vector)families.lastElement()).get(i)).get(1)).
        equals("N,N")   ){
          ASIn.add((int[])((Vector)((Vector)families.lastElement()).get(i)).
          get(3));
          ASIn.add((int[])((Vector)((Vector)families.lastElement()).get(i)).
          get(4));
        }
      }
      //
      noASCtrls = ASIn.size() - 3;
      //write it to file
      try{
        BufferedWriter bwAS = new BufferedWriter(new FileWriter(infolder +
        "ASIn.csv"));
        for (int i=0;i<((int[])ASIn.get(0)).length;i++){
          for (int j=0;j<ASIn.size();j++){
            bwAS.write(((int[])ASIn.get(j))[i] + ",");
          }
          bwAS.newLine();
        }
        bwAS.close();
      } catch (Exception e){
        System.out.println("Error writing ASIn.csv: " + e);
      }
    }
    //clean up
    families.removeAllElements();
  }
  //clean up
  affectedPerson.removeAllElements();
  minPerson.removeAllElements();
  prevGenHT.clear();
  peopleHT.clear();
  peopleHT_.clear();
  generationsHT.clear();
}

/*******
*
*/
public Vector makeMarriedIn(int[] sHap1, int[] sHap2){
  Vector minPerson = new Vector();
  minPerson.add("N,N");//parentID (N is no parent)
  minPerson.add("U");//disease status
  minPerson.add(sHap1);//haplotpye1
  minPerson.add(sHap2);//haplotype2
  return minPerson;
}
public Vector sort(Enumeration e){
  Vector v = new Vector();
  String st = (String)e.nextElement();
  v.add(new Integer(st));
  while (e.hasMoreElements()){
    st = (String)e.nextElement();
    int x = (new Integer(st)).intValue();
```

```
      //go through each no allready added
      boolean added = false;
      for (int i=0;i<v.size();i++){
        //if x is >, add in fron
        if(x<((Integer)v.get(i)).intValue()){
           v.insertElementAt(new Integer(x),i);
           added = true;
           break;
        }
      }
      if (!added)
        v.add(new Integer(x));
    }
    return v;
  }

  /**
   *
   */
  public Vector getChildChr(int[] chr0, int[] chr1, String parentDS, String key){
    String ds = "";
    int[] childChr = new int[chr0.length];
    double rand = Math.random();
    int chr2Use = 0;
    if(rand<=0.5){//start getting chr1 from parent1
      //set chr2Use to 0
      chr2Use = 0;
    } else {
      chr2Use = 1;
    }
    //go through each SNP
    for (int k=0;k<chr0.length;k++){
      //add the markers, one by one
      if(chr2Use==0){
        childChr[k] = chr0[k];
        if(k==mutMarker)
          ds=parentDS;
      }else if (chr2Use==1){
        childChr[k] = chr1[k];
        if(k==mutMarker)
          ds="U";
      }
      //test for recombination
      //get rand no
      rand = Math.random();
      //if rand<(1x10(8)*dbm) then we have a recombination event
      if(rand < (recombRate*distBetweenMarkers)){
        //and we change parental chromosomes
        if(chr2Use==0)
          chr2Use=1;
        else if(chr2Use==1)
          chr2Use=0;
      }
    }
    Vector v = new Vector();
    v.add(childChr);
    v.add(ds);
    return v;
  }
  //use these to tell the main method whether or not the simulator has generated
  //>=3 multi generational familes
  public void setStatus(boolean s){
    if(s)
      status = true;
    else
      status = false;
  }
  public boolean getStatus(){
    return status;
  }

  //
  public int getASNoCtrls(){
    return noASCtrls;
  }
}
```

```
class SummarStats{
  //
  private String infolder = "//net//uisdein//export//usr0//s9636861//code//sim2//";
  //
  public SummarStats(){
  //get average stats for family size etc...
  String stNoASPerms = "";
  String stNoSims = "";
  try{
    BufferedReader brStats = new BufferedReader(new FileReader(infolder +
    "AllStatsSummary.txt"));
    brStats.readLine();//ignore line
    stNoSims = brStats.readLine();//line contains noSims
    //remove text from line
    stNoSims = stNoSims.replaceFirst("Number of simulations:","");
    brStats.readLine();//ignore line
    stNoASPerms = brStats.readLine();//line contains noASPerms
    //remove text from line
    stNoASPerms = stNoASPerms.replaceFirst("Number of AS permutations:","");
    brStats.close();
  } catch (Exception e){
    System.out.println("Exception reading from stats file: " + e);
  }
  //read in general info regarding sim run...
  double noPerms = (new Double(stNoASPerms)).doubleValue();
  double noSims = (new Double(stNoSims)).doubleValue();
  //for each sim folder
  int i=0;
  int truePos = 0;
  int falsePos = 0;
  Vector mutRegPValues = new Vector();
  Vector otherPValues = new Vector();
  Vector mutRegBlocks = new Vector();
  Vector otherBlocks = new Vector();
  Vector famDists = new Vector();
  String statsFile = infolder + "sim" + i + "//StatsSummary.txt";
  String distMatFile = infolder + "sim" + i + "//dist.matrix";
  String statsFolder = infolder + "sim" + i;
  while ((new File(statsFile)).exists()){
    //get stats
    try{
      BufferedReader br = new BufferedReader(new FileReader(statsFile));
      String line = br.readLine();
      line = br.readLine();
      while (line!=null){
        StringTokenizer tok = new StringTokenizer(line,",");
        tok.nextToken();
        double p = (new Double(tok.nextToken())).doubleValue()/noPerms;
        String isMutReg = tok.nextToken();
        String noBlocks = tok.nextToken();
        //if we are looking at the mut carrying region
        if(isMutReg.equals("1")){
          if(p <= 0.05){
            truePos++;
            mutRegPValues.add( (new Double(p)).toString() );
            mutRegBlocks.add(noBlocks);
          }
        //else we are looking at a false region
        } else {
          if(p <= 0.05){
            falsePos++;
            otherPValues.add( (new Double(p)).toString() );
            otherBlocks.add(noBlocks);
          }
        }
        line = br.readLine();
      }
      br.close();
    }
    catch (Exception e){
      System.out.println("exception reading from file " + i + ": " + e);
    }
    //also get info from each dist.matrix file
    try{
      BufferedReader br = new BufferedReader(new FileReader(distMatFile));
      br.readLine();
```

```
      br.readLine();
      String dist1 = br.readLine();
      dist1 = dist1.replaceFirst(".*: ","");
      famDists.add(dist1);
      String dist2 = br.readLine();
      dist2 = dist2.replaceFirst(".*: ","");
      famDists.add(dist2);
      String dist3 = br.readLine();
      dist3 = dist3.replaceFirst(".*: ","");
      famDists.add(dist3);
      br.close();
    }  catch (Exception e){
      System.out.println("exception reading from file " + i + ": " + e);
    }
    //delete folders
    File[] content = (new File(statsFolder)).listFiles(new FileFilter(){
      public boolean accept(File f){
        return true;
      }
    } );
    for (int j=0;j<content.length;j++){
      boolean fot = content[j].delete();
      if(content[j].isDirectory()){
        File[] xxx = content[j].listFiles(new FileFilter(){
          public boolean accept(File f){
            return true;
        }} );
          }
        }
        //get next folder
        i++;
        statsFile = infolder + "sim" + i + "//StatsSummary.txt";
        distMatFile = infolder + "sim" + i + "//dist.matrix";
        statsFolder = infolder + "sim" + i;
      }//END OF WHILE EACH STATSTFOLDER.EXISTS
      //calculate the average of the fam distances
      double avFamDist = 0.0;
      for (int j=0;j<famDists.size();j++){
        avFamDist += (new Double(  (String)famDists.get(j) )).
        doubleValue();
      }
      avFamDist = avFamDist/famDists.size();
      //calculate average true pos scores
      double avTruePosP = 0.0;
      if(mutRegPValues.size()==0)
        avTruePosP = Double.NaN;
      else {
        for (int j=0;j<mutRegPValues.size();j++){
          avTruePosP += (new Double(  (String)mutRegPValues.get(j) )).
          doubleValue();
        }
        avTruePosP = avTruePosP/mutRegPValues.size();
      }
      //calculate average false pos scores
      double avFalsePosP = 0.0;
      if(otherPValues.size()==0)
      avFalsePosP = Double.NaN;
      else {
    for (int j=0;j<otherPValues.size();j++){
      avFalsePosP += (new Double(  (String)otherPValues.get(j) )).doubleValue();
    }
      avFalsePosP = avFalsePosP/otherPValues.size();
    }
    //calculate average true pos blocks
    double avTruePosBlocks = 0.0;
    if(mutRegBlocks.size()==0)
    avTruePosBlocks = Double.NaN;
    else {
  for (int j=0;j<mutRegBlocks.size();j++){
    avTruePosBlocks += (new Double(  (String)mutRegBlocks.get(j) )).doubleValue();
  }
  avTruePosBlocks = avTruePosBlocks/mutRegBlocks.size();
}
//calculate average false pos scores
double avFalsePosBlocks = 0.0;
if(otherBlocks.size()==0)
avFalsePosBlocks = Double.NaN;
```

```
  else {
    for (int j=0;j<otherBlocks.size();j++){
      avFalsePosBlocks += (new Double(  (String)otherBlocks.get(j) )).doubleValue();
    }
    avFalsePosBlocks = avFalsePosBlocks/otherBlocks.size();
  }

        //append the results to a AllStatsSummary
        try {
          BufferedWriter bw = new BufferedWriter(new FileWriter((infolder +
          "AllStatsSummary.txt"),true));
          bw.newLine();
          //summary of families
          bw.write("Average distance between families (generations):" + avFamDist);
          bw.newLine();
          //summary of stats
          bw.newLine();
          bw.write("True positives: " + truePos + " (" + ((double)(truePos/noSims)*
          100.0) + "%)");
          bw.newLine();
          bw.write("False positives: " + falsePos);
          bw.newLine();
          bw.write("P values for true positives: ");
          bw.write("" + avTruePosP + " (");
          if(mutRegPValues.size()>0){
            bw.write("" + mutRegPValues.get(0));
            for(int j=1;j<mutRegPValues.size();j++){
              bw.write("," + mutRegPValues.get(j));
            }
          }
          bw.write(")");
          bw.newLine();
          bw.write("P values for false positives: ");
          bw.write("" + avFalsePosP + " (");
          if(otherPValues.size()>0){
            bw.write("" + otherPValues.get(0));
            for(int j=1;j<otherPValues.size();j++){
              bw.write("," + otherPValues.get(j));
            }
          }
          bw.write(")");
          bw.newLine();
          bw.write("Size in blocks for true positives: ");
          bw.write("" + avTruePosBlocks + " (");
          if(mutRegBlocks.size()>0){
            bw.write("" + mutRegBlocks.get(0));
            for(int j=1;j<mutRegBlocks.size();j++){
          bw.write("," + mutRegBlocks.get(j));
        }
      }
      bw.write(")");
      bw.newLine();
      bw.write("Size in blocks for false positives: ");
      bw.write("" + avFalsePosBlocks + " (");
      if(otherBlocks.size()>0){
        bw.write("" + otherBlocks.get(0));
        for(int j=1;j<otherBlocks.size();j++){
          bw.write("," + otherBlocks.get(j));
        }
      }
      bw.write(")");
      //close file
      bw.close();
    } catch (Exception e){
      System.out.println("Exception writing to AllStatsSummary: " + e);
    }
  }
}

class FindSigRegions {

  public FindSigRegions(String ASInPath, String ASOutPath, String ASStatsPath,
  String blockDefPath, String datPath, int mutMarkerIndex) {
    Vector col0 = new Vector();
    Vector col1 = new Vector();
    Vector col2 = new Vector();
    Vector ASScores = new Vector();
```

```
Vector ASPValues = new Vector();
Vector blockNos = new Vector();
//1
//read the ASIn file -> col 0,1,2
//store the first three columns
try {
  BufferedReader br = new BufferedReader(new FileReader(ASInPath));
  String line = br.readLine();
  while (line != null){
    StringTokenizer tok = new StringTokenizer(line,",");
    col0.add(tok.nextElement());
    col1.add(tok.nextElement());
    col2.add(tok.nextElement());
    line = br.readLine();
  }
  br.close();
} catch (Exception e){
  System.out.println("Exception in fsr import ASIn.csv: " + e);
}

//2
//now get the AS.in scores -> ASScores
try{
  BufferedReader br2 = new BufferedReader(new FileReader(ASOutPath));
  String line = br2.readLine();
  while (line!=null){
    StringTokenizer tok = new StringTokenizer(line,",");
    tok.nextElement();//ignore 1st element
    tok.nextElement();//ignore 2nd
    ASScores.add(tok.nextElement());//store the 3rd as the AS score
    line = br2.readLine();
  }
  br2.close();
} catch (Exception e){
  System.out.println("Exception in fsr import AS.in: " + e);
}

//3
//now read in the p values from AS_permutations.stats -> ASPValues
try {
  BufferedReader br3 = new BufferedReader(new FileReader(ASStatsPath));
  String line = br3.readLine();
  int count = 0;
  while (line!=null){
    if(count>11)
      ASPValues.add(line);
    count++;
    line = br3.readLine();
  }
  br3.close();
} catch (Exception e){
  System.out.println("Exception in fsg import AS_permutations.stats: " + e);
}

//4
//read in file with block no for each marker -> blockNos
try {
  BufferedReader br4 = new BufferedReader(new FileReader(blockDefPath));
  String line = br4.readLine();
  while (line!=null){
    StringTokenizer tok = new StringTokenizer(line,"\t");
    tok.nextElement();//ignore 1st element
    tok.nextElement();//ignore 2nd
    blockNos.add(tok.nextElement());//store the 3rd as the block number
    line = br4.readLine();
  }
  br4.close();
} catch (Exception e){
  System.out.println("Exception in fsg import MyHapMap.out: " + e);
}

//2 + 3 + 4
//and now, go through all the ASIn cols and where there are sequences
//of sharing, calculate the average scores and p values
double rSum = 0.0;//sum of scores
double pSum = 0.0;//sum of p values
int count = 0;
```

```
String lastBlockNo = "";
int numBlocks = 0;
Vector blockScore = new Vector();
Vector blockP = new Vector();
Vector blockBlocks = new Vector();
int mutBlock = 100;
boolean mutBlockActive = false;
//so, for each marker
for (int i=0;i<col0.size();i++){
  //if all markers are equal store the score and p values
  if ( ((String)col0.get(i)).equals((String)col1.get(i)) && ((String)col0.get(i)).
  equals((String)col2.get(i))){
    //sum the score and p vlaues
    rSum += (new Double((String)ASScores.get(i))).doubleValue();
    pSum += (new Double((String)ASPValues.get(i))).doubleValue();
    count++;
    //++ the no of blocks if it is a new block
    if( ((String)blockNos.get(i)).equals("null") )
      numBlocks++;
    else if (  !((String)blockNos.get(i)).equals(lastBlockNo) )
      numBlocks++;
    //if it crosses the mutation, then mark this block as true
    if(i==mutMarkerIndex){
      mutBlockActive = true;
    }
    //if this is the last marker in the column, and it is shared, then we need to
    //add that as a shared block as well
    if (i == col0.size() && count > 1  && ((pSum/count) > 0.949)){
      blockScore.add(new Double(rSum/count));
      blockP.add(new Double(pSum/count));
      blockBlocks.add(new Integer(numBlocks));
      if(mutBlockActive){
        mutBlock = blockScore.size() - 1;
        mutBlockActive = false;
      }
    }
  }
  //else if there is no sharing
  else{
    //if previous sharing >1, we store the last blocks results
    if(count > 1  && ((pSum/count) > 0.949) ){
      blockScore.add(new Double(rSum/count));
      blockP.add(new Double(pSum/count));
      blockBlocks.add(new Integer(numBlocks));
      if(mutBlockActive){
        mutBlock = blockScore.size() - 1;
        mutBlockActive = false;
      }
    }
    rSum = 0.0;
    pSum = 0.0;
    numBlocks = 0;
    count = 0;
  }
}
//add the last block in it is still being shared...
if(count > 1  && ((pSum/count) > 0.949)  ){
  blockScore.add(new Double(rSum/count));
  blockP.add(new Double(pSum/count));
  blockBlocks.add(new Integer(numBlocks));
  if(mutBlockActive){
    mutBlock = blockScore.size() - 1;
  }
}

//5
//now call NewASPermAnalysis to calculate the modified p values for each
//region
//now generate a new data.dat file for each of these regions
//first read in the present file
Vector lines = new Vector();
try {
  BufferedReader br5 = new BufferedReader(new FileReader(datPath));
  String line = br5.readLine();
  while (line!=null){
    lines.add(line);
    line = br5.readLine();
```

```
      }
      br5.close();
    } catch (Exception e){
      System.out.println("Exception in fsg import AS_permutations.stats: " + e);
    }
    //and then re-write with the final two lines altered
    //for each of the regions...
    for (int i=0;i<blockScore.size();i++){
      File dir = new File((new File(ASInPath)).getParent() + "//sr" + i);
      dir.mkdirs();
      String datPathTemp = (new File(ASInPath)).getParent() + "//sr" + i +
      "//data.dat";
      try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(datPathTemp));
        bw.write((String)lines.get(0));
        bw.newLine();
        bw.write((String)lines.get(1));
        bw.newLine();
        bw.write(((Integer)blockBlocks.get(i)).toString());
        bw.newLine();
        bw.write(((Double)blockScore.get(i)).toString());
        bw.close();
      } catch (Exception e){
        System.out.println("Exception in fsg writing to data.dat: " + e);
      }
      //identify the block that contains the 'real' mutation
      if(i==mutBlock){
        File fl = new File(  (new File(ASInPath)).getParent() + "//sr" + i +
        "//MUTATION");
        try {
          fl.createNewFile();
        } catch (Exception e){
          ;
        }
      }
    }
  }
}
```

Appendix B: Papers arising from this thesis

Le Hellard S, Lee AJ, Underwood S, Thomson PA, Morris SW, Torrance HS, Anderson SM, Adams RR, Navarro P, Christoforou A et al. (2006) Haplotype Analysis and a Novel Allele-sharing Method Refines a Chromosome 4p Locus Linked to Bipolar Affective Disorder. Biol Psychiatry 61(6):797-805

Lee AJ, Thomson PA, Le Hellard S, Adams R, Muir WJ, Blackwood DHR, Wray NR, Férec C, Porteous DJ, Evans KL (2008) An Allele Sharing Method for Fine Mapping Linkage Loci: Proof of Principle and Application to Bipolar Disorder. Submitted for publication